



ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Informática

SiteUp: plataforma para la vigilancia de
la disponibilidad de servicios de Internet

José Tomás Tocino García

Cádiz, 14 de mayo de 2014



ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Informática

SiteUp: plataforma para la vigilancia de
la disponibilidad de servicios de Internet

DEPARTAMENTO: Ingeniería Informática.
DIRECTOR DEL PROYECTO: Iván Ruiz Rube
AUTOR DEL PROYECTO: José Tomás Tocino García.

Cádiz, 14 de mayo de 2014

Fdo.: José Tomás Tocino García

Este documento se halla bajo la licencia FDL (Free Documentation License). Según estipula la licencia, se muestra aquí el aviso de copyright. Se ha usado la versión inglesa de la licencia, al ser la única reconocida oficialmente por la FSF (Free Software Foundation).

Copyright © 2014 José Tomás Tocino García.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Agradecimientos

Quiero agradecer y dedicar la presente memoria y, por extensión, el proyecto SiteUp al completo:

- A mi familia y mi pareja, que igualmente como ocurrió durante el desarrollo del PFC de la Ingeniería Técnica allá en 2011, han tenido que *soportar* el desarrollo de este PFC durante bastante tiempo.
- A Manuel Palomo, Rafael R. Galván, Juan M. Dodero e Iván Ruiz. Gracias a ellos he formado parte de la UCA como becario, como alumno colaborador y como personal investigador desde 2009 hasta 2013, un periodo en el que he aprendido más de lo imaginable.
- A mis compañeros más cercanos de la carrera, que me han apoyado y de los que me he nutrido para desarrollar este proyecto.
- A Guido van Rossum por crear Python, un lenguaje precioso que hace fácil hasta las tareas más complejas y pone de manifiesto las obvias limitaciones de otros lenguajes más populares como PHP. Sin Python, este proyecto no sería nada.
- A los creadores de Django, por desarrollar una pieza de software tan lógica y a la vez tan potente y útil.
- A los que apuestan por el software libre y el conocimiento abierto. Sus herramientas, bibliotecas y documentos han hecho posible que SiteUp llegase a buen puerto.

Resumen

SiteUp es un proyecto para la vigilancia de la disponibilidad de servicios de Internet. Consta de una plataforma web, en la que los usuarios tienen la posibilidad dar de alta *chequeos* de varios tipos: envío de pings, chequeo de puertos, chequeo de registros DNS y comprobaciones mediante peticiones HTTP. Estos chequeos son ejecutados por el sistema de forma periódica y generan notificaciones cuando detectan fallos. Estas notificaciones se envían mediante correo electrónico o a través de SiteUp Client, una aplicación Android desarrollada a tal efecto. Los usuarios tienen también la posibilidad de revisar la información obtenida de los chequeos a través de la web a lo largo del tiempo.

Palabras clave: Internet, Web, Servicio, SaaS, Vigilancia, Disponibilidad, Monitorización, Ping, DNS, HTTP, Puerto, Servidor, Python, Django, Celery, RabbitMQ.

Índice general

Índice general	11
Índice de figuras	17
Índice de tablas	21
1. Introducción	23
1.1. Contexto social	23
1.2. Motivación	23
1.3. Objetivos	24
1.3.1. Funcionales	24
1.3.2. Transversales	25
1.4. Estado del arte	25
1.4.1. Métodos de monitorización	25
1.4.2. Servicios similares	30
1.5. Alcance	31
1.5.1. Licencia	33
1.6. Estructura del documento	34
1.7. Acrónimos	34
2. Planificación	37
2.1. Metodología de desarrollo	37
2.1.1. Primera iteración: adquisición de conocimientos y elicitación de requisitos	37
2.1.2. Segunda iteración: desarrollo de módulo de herramientas básicas de chequeo	38
2.1.3. Tercera iteración: inicio de proyecto Django y CRUD básico	38
2.1.4. Cuarta iteración: integración del motor de tareas asíncronas	39
2.1.5. Quinta iteración: desarrollo de la aplicación Android	39
2.2. Planificación temporal	39
2.3. Organización	39
2.4. Recursos inventariables	41
2.5. Costes	42
2.6. Riesgos	42
2.6.1. Compromiso de la plataforma de despliegue	42
2.6.2. Limitación de recursos	43

2.6.3. Rechazo de los servicios vigilados	43
3. Requisitos del sistema	45
3.1. Objetivos del sistema	45
3.2. Catálogo de requisitos	46
3.2.1. Requisitos funcionales	46
3.2.2. Requisitos de información	49
3.2.3. Requisitos no funcionales	52
3.3. Alternativas de solución	53
3.3.1. Frameworks para el desarrollo de la plataforma web	54
3.3.2. Sistemas de gestión de bases de datos	54
3.3.3. Sistemas para la ejecución asíncrona de tareas	55
3.3.4. Plataformas para el desarrollo de la aplicación móvil	55
4. Análisis	59
4.1. Modelo conceptual	59
4.1.1. Usuario	59
4.1.2. Grupo de chequeos	61
4.1.3. Chequeo	61
4.1.4. Registro de chequeo	61
4.1.5. Estado de chequeo	62
4.1.6. Chequeo - Ping	62
4.1.7. Chequeo - Port	63
4.1.8. Chequeo - DNS	63
4.1.9. Chequeo - HTTP	63
4.2. Modelo de casos de uso	63
4.2.1. Actores	64
4.2.2. Subsistema de gestión de usuarios	64
4.2.3. Subsistema de gestión de grupos de chequeos	66
4.2.4. Subsistema de gestión de chequeos	69
4.3. Modelo de interfaz de usuario	72
4.3.1. Modelo de navegación	72
4.3.2. Prototipos visuales de la interfaz web	74
4.3.3. Prototipos visuales de la aplicación móvil	75
5. Diseño	83
5.1. Arquitectura del sistema	83
5.1.1. Arquitectura física	83
5.1.2. Arquitectura lógica	84
5.2. Diseño detallado de componentes	91
5.2.1. Lanzamiento y notificación de un chequeo	91
5.3. Diseño físico de datos	94
5.3.1. User	94
5.3.2. UserExtra	94
5.3.3. CheckGroup	97
5.3.4. BaseCheck	97
5.3.5. DnsCheck	98

5.3.6.	HttpCheck	98
5.3.7.	PortCheck	98
5.3.8.	PingCheck	99
5.3.9.	CheckLog	99
5.3.10.	CheckStatus	100
5.3.11.	Session	100
5.3.12.	MigrationHistory	100
5.3.13.	ContentType	101
5.4.	Diseño de la imagen corporativa	101
5.4.1.	Diseño del logotipo	101
5.4.2.	Elección de la gama cromática	102
5.4.3.	Elección de la tipografía	103
5.5.	Diseño de la interfaz de usuario de la plataforma web	103
5.5.1.	Pantalla de inicio	103
5.5.2.	Pantalla de inicio de sesión	105
5.5.3.	Pantalla de recuperación de contraseña	105
5.5.4.	Pantalla de registro de usuario	106
5.5.5.	Pantalla de lista de chequeos	107
5.5.6.	Pantalla de detalles de chequeo	108
5.5.7.	Pantalla de creación de grupo	108
5.5.8.	Pantallas de creación de chequeo	108
5.5.9.	Pantalla genérica de borrado	110
5.5.10.	Pantalla del perfil de usuario	110
5.6.	Diseño de la interfaz de usuario de la aplicación móvil	113
5.6.1.	Aspecto desde el lanzador	113
5.6.2.	Pantalla de carga	114
5.6.3.	Pantalla de inicio de sesión	114
5.6.4.	Pantalla de listado de chequeos	114
5.6.5.	Notificación	116
6.	Implementación	119
6.1.	Entorno de construcción	119
6.1.1.	Aplicación web	119
6.1.2.	Aplicación móvil	124
6.2.	Organización del código fuente	125
6.2.1.	Plataforma web	126
6.2.2.	Aplicación móvil	128
6.3.	Detalles de implementación	129
6.3.1.	Implementación de los chequeos	129
6.3.2.	Uso de herencia de modelos	132
6.3.3.	Workflow de front-end	133
6.4.	Gestión de la base de datos	137
6.4.1.	Cambios en la estructura	139
7.	Pruebas	141
7.1.	Estrategia	141
7.2.	Entorno de pruebas	141

7.3. Roles	142
7.3.1. Desarrollador principal	142
7.3.2. Probadores externos	142
7.4. Pruebas funcionales	142
7.4.1. Pruebas de sistema	143
7.4.2. Pruebas de aceptación	144
7.5. Pruebas no funcionales	144
7.5.1. Pruebas de seguridad	144
7.5.2. Pruebas de fiabilidad	145
7.5.3. Pruebas de accesibilidad	145
7.5.4. Verificación de la calidad del código	145
8. Conclusiones	149
8.1. Objetivos cumplidos	149
8.1.1. Limitaciones del proyecto	150
8.2. Conclusiones personales	150
8.2.1. Lecciones aprendidas	150
8.3. Difusión y reconocimiento	151
8.4. Trabajo futuro	151
8.4.1. SiteUp Probes	152
8.4.2. Monetización	152
8.4.3. Ampliación de los tipos de chequeos	152
8.4.4. Ampliación de tipos de notificación	153
A. Manual de instalación de la plataforma web	155
A.1. Instalación inicial	156
A.1.1. Descarga de código	156
A.1.2. Entorno virtual y dependencias	156
A.1.3. Instalación de credenciales	157
A.1.4. Creación de la base de datos	157
A.1.5. Disposición de ficheros estáticos	157
A.1.6. Instalación del servidor	157
A.1.7. Instalación del bróker de mensajes	158
A.1.8. Ejecución de los servicios	158
A.1.9. Configuración del servidor frontal	158
A.1.10. Lanzamiento final	159
A.2. Pruebas de implantación	160
A.3. Problemas habituales	161
A.3.1. Activación de nginx	161
A.3.2. Mejora del rendimiento	161
B. Manual de usuario	163
B.1. Introducción a SiteUp	163
B.2. Requisitos previos	164
B.3. Uso de la plataforma web	164
B.3.1. Acceso	164
B.3.2. Registro de usuario	164

B.3.3. Inicio de sesión	165
B.3.4. Recuperación de contraseña	166
B.3.5. Pantalla de listado de chequeos	167
B.3.6. Cerrar sesión	167
B.3.7. Edición de datos de usuario	169
B.3.8. Creación de grupo de chequeos	169
B.3.9. Operaciones con grupos de chequeos	170
B.3.10. Creación de chequeo	171
B.3.11. Operaciones con chequeos	176
B.3.12. Recepción de notificaciones por correo electrónico	179
B.4. Uso de la aplicación móvil	180
B.4.1. Instalación	180
B.4.2. Lanzamiento	180
B.4.3. Carga de la aplicación	182
B.4.4. Inicio de sesión	182
B.4.5. Revisión de chequeos	184
B.4.6. Recepción de notificaciones	184
C. GNU Free Documentation License	187
1. APPLICABILITY AND DEFINITIONS	187
2. VERBATIM COPYING	189
3. COPYING IN QUANTITY	189
4. MODIFICATIONS	190
5. COMBINING DOCUMENTS	192
6. COLLECTIONS OF DOCUMENTS	192
7. AGGREGATION WITH INDEPENDENT WORKS	192
8. TRANSLATION	193
9. TERMINATION	193
10. FUTURE REVISIONS OF THIS LICENSE	194
11. RELICENSING	194
ADDENDUM: How to use this License for your documents	195
Bibliografía y referencias	197

Índice de figuras

2.1. Diagrama de Gantt	40
4.1. Modelo conceptual	60
4.2. Diagrama de actores	64
4.3. Casos de uso del subsistema de gestión de usuarios	67
4.4. Casos de uso del subsistema de gestión de grupos de chequeos	67
4.5. Casos de uso del subsistema de gestión de chequeos	69
4.6. Modelo de navegación de la aplicación web	73
4.7. Modelo de navegación de la aplicación Android	74
4.8. Home	75
4.9. Login	76
4.10. Recuperación de contraseña	76
4.11. Registro de usuario	77
4.12. Dashboard	77
4.13. Detalle de chequeo	78
4.14. Creación de grupo de chequeos	78
4.15. Formulario genérico de creación de chequeo	79
4.16. Pantalla de carga	79
4.17. Pantalla de login	80
4.18. Pantalla de listado de chequeos	80
4.19. Pantalla de notificación	81
5.1. Arquitectura lógica del sistema web	85
5.2. Arquitectura lógica de la aplicación Android	89
5.3. Diagrama de secuencia del proceso de lanzamiento de un chequeo	92
5.4. Diseño de la base de datos (I)	95
5.5. Diseño de la base de datos (II)	96
5.6. Logotipo de SiteUp	102
5.7. Detalle de la paleta de colores de SiteUp	102
5.8. Muestra de Open Sans en diferentes pesos	103
5.9. Pantalla de inicio	104
5.10. Pantalla de inicio de sesión	105
5.11. Pantalla de recuperación de contraseña	106
5.12. Pantalla de registro de usuario	106
5.13. Pantalla de lista de chequeos	107
5.14. Pantalla de detalles de chequeo	109

5.15. Pantalla de creación de grupo de chequeos	109
5.16. Pantalla de elección de tipo de chequeo	110
5.17. Pantalla de introducción de datos del chequeo	111
5.18. Pantalla genérica de borrado	111
5.19. Pantalla del perfil de usuario	112
5.20. Icono de SiteUp en el lanzador	113
5.21. Pantalla de carga de la aplicación móvil	114
5.22. Pantalla de inicio de sesión de la aplicación móvil	115
5.23. Pantalla de chequeos de la aplicación móvil	115
5.24. Aspecto de las notificaciones	117
6.1. Diagrama de flujo de trabajo del front-end	134
6.2. Diagrama ER de ejemplo 1	137
7.1. Plataforma web vista mediante un Apple iPad 2	146
7.2. Plataforma web vista mediante un Jiayu G2 (orientación horizontal)	146
7.3. Plataforma web vista mediante un Galaxy Nexus	147
8.1. Logotipo VIII CUSL	151
B.1. Botones principales para el usuario anónimo	164
B.2. Formulario de registro de usuario	165
B.3. Mensaje indicando el registro correcto	165
B.4. Formulario de inicio de sesión	166
B.5. Formulario de recuperación de contraseña	166
B.6. Pantalla de confirmación de envío de código de reinicio	167
B.7. Pantalla para la introducción de la nueva contraseña	167
B.8. Panel de control	168
B.9. Botones principales para el usuario logueado	168
B.10. Pantalla de edición del perfil de usuario	169
B.11. Botón para crear nuevo grupo de chequeos	170
B.12. Formulario de creación de grupo de chequeos	170
B.13. Menú de acciones de grupo	170
B.14. Formulario de confirmación de borrado	171
B.15. Pantalla de selección del tipo de chequeo a crear	172
B.16. Formulario de creación de chequeo por ping	172
B.17. Formulario de creación de chequeo de puerto	174
B.18. Formulario de creación de chequeo de registro DNS	175
B.19. Formulario de creación de chequeo por HTTP	176
B.20. Aspecto de un chequeo válido	176
B.21. Aspecto de un chequeo en estado de error	177
B.22. Pantalla de detalle de chequeo	178
B.23. Email de notificación de cambio de estado	180
B.24. Email diario de resumen	181
B.25. Icono de SiteUp en el lanzador	182
B.26. Pantalla de carga de la aplicación móvil	183
B.27. Pantalla de inicio de sesión de la aplicación móvil	183

B.28.Pantalla de chequeos de la aplicación móvil	184
B.29.Aspecto de las notificaciones	185

Índice de tablas

1.1. Tabla comparativa de servicios de monitorización	32
2.1. Comparación de la estimación con los tiempos reales	39
2.2. Tabla de planificación de costes	42
3.1. OBJ-1, objetivo del sistema 1	45
3.2. OBJ-2, objetivo del sistema 2	46
3.3. RQF-1, requisito funcional 1: creación de cuenta de usuario	46
3.4. RQF-2, requisito funcional 2: edición de cuenta de usuario	46
3.5. RQF-3, req. funcional 3: creación de grupo de chequeos	47
3.6. RQF-4, req. funcional 4: edición de grupo de chequeos	47
3.7. RQF-5, req. funcional 5: eliminación de grupo de chequeos	47
3.8. RQF-6, req. funcional 6: activación de grupo de chequeos	47
3.9. RQF-7, req. funcional 7: creación de chequeos	47
3.10. RQF-8, req. funcional 8: actualización de chequeos	48
3.11. RQF-9, req. funcional 9: eliminación de chequeos	48
3.12. RQF-11, req. funcional 11: activación de chequeos	48
3.13. RQF-11, req. funcional 11: revisión de chequeos	48
3.14. RQF-12, req. funcional 12: notificación por correo	48
3.15. RQF-13, req. funcional 13: lanzamiento de chequeos	49
3.16. RQF-14, req. funcional 14: mantenimiento de registros	49
3.17. RQF-15, req. funcional 15: listado de chequeos en app	49
3.18. RQF-16, req. funcional 16: notificaciones mediante la app	49
3.19. IRQ-1, req. de información 1: usuario	50
3.20. IRQ-2, req. de información 2: chequeo Ping	50
3.21. IRQ-3, req. de información 3: chequeo DNS	50
3.22. IRQ-4, req. de información 4: chequeo de puerto	51
3.23. IRQ-5, req. de información 5: chequeo HTTP	51
3.24. IRQ-6, req. de información 6: registro de chequeo	51
3.25. IRQ-7, req. de información 7: estado de chequeo	52
3.26. IRQ-8, req. de información 8: grupo de chequeos	52
3.27. NRQ-1, req. no funcional 1: cifrado de contraseñas	52
3.28. NRQ-2, req. no funcional 2: control de acceso	52
3.29. NRQ-3, req. no funcional 3: ejecución de chequeos	53
3.30. NRQ-4, req. no funcional 4: adaptación a dispositivos	53
3.31. NRQ-5, req. no funcional 5: monitorización de la ejecución	53

3.32. Cuota de mercado de plataformas móviles en enero de 2014	57
--	----

Capítulo 1

Introducción

1.1. Contexto social

Las tecnologías de la información en general e Internet en particular son ya parte integral de la sociedad. Casi todos los ámbitos de la vida, desde las interacciones sociales hasta la búsqueda de empleo, cuentan ya con su reflejo en las tecnologías de la información, a menudo mediante el uso de servicios web a través de Internet.

No solo los aspectos tradicionales de la sociedad tienen su presencia en las redes, también han surgido nuevos modelos empresariales propios de Internet que han crecido de manera importante y se han situado a niveles comparables a los de las empresas tradicionales. Empresas puramente digitales como Facebook o Twitter ya cotizan en bolsa y realizan operaciones bursátiles del orden de miles de millones de dólares [17].

Se pone pues de manifiesto la importancia de la fiabilidad de los servicios e infraestructuras de los que dependen estos nuevos modelos de negocio. El *uptime* – un término inglés que describe el porcentaje de tiempo que un servicio se mantiene disponible – debe ser siempre cercano al 100 %, dado que en caso contrario los potenciales usuarios del servicio se encontrarán con que no pueden acceder a él, dando lugar incluso a pérdidas económicas. Es el caso de Amazon, que llegó a perder 4.8 millones de dólares al sufrir un fallo que dejó inaccesible su web durante 40 minutos [2].

De todo lo expuesto se extrae la necesidad de contar con sistemas para monitorizar la disponibilidad de estos servicios y, en caso necesario, actuar de manera que puedan subsanarse las causas de los problemas.

1.2. Motivación

La idea de desarrollar este proyecto surge a raíz de una necesidad personal del desarrollador del proyecto.

En octubre de 2013 tomé parte en un importante proceso de selección en el que las comunicaciones se estaban llevando a cabo a través de correo electrónico. En particular, la cuenta de correo electrónico que estaba usando (info@josetomastocino.com) tenía un dominio personalizado y estaba gestionada a través de Google Apps [21]. Esto era posible gracias a que en el dominio se dieron de alta unos registros de tipo MX y TXT que hacían que el correo se redirigiese a los servidores de Google.

A mitad del proceso de selección, la empresa que gestionaba el dominio de mi web personal (y, por extensión, mi correo electrónico) tuvo un problema y reemplazó los registros DNS personalizados por unos por defecto. En particular, los registros MX y TXT quedaron en blanco, y los registros A apuntaron a una página de *párking*¹, de forma que los chequeos de tipo ping que tenía puestos no dieron error, ya que efectivamente el servidor devolvía el ping, pero el contenido de los registros no era el correcto, ni los correos se estaban direccionando bien.

Esta situación se prolongó durante varios días, en los cuales perdí varios mensajes importantes. Esto podría haberse detectado rápidamente si hubiera tenido alguna clase de vigilancia que verificase que los registros DNS tenían el contenido correcto.

De ahí nace la necesidad de realizar un estudio de las alternativas existentes para este tipo de vigilancia y, al concluir que no existía ninguna alternativa libre, se decide iniciar el proyecto SiteUp para la vigilancia de servicios de Internet.

1.3. Objetivos

A la hora de definir los objetivos de un sistema, podemos agruparlos en dos tipos diferentes: **funcionales** y **transversales**. Los primeros se refieren a *qué* debe hacer la aplicación que vamos a desarrollar, e inciden directamente en la experiencia del usuario y de potenciales desarrolladores.

Por otro lado, los objetivos transversales son aquellos invisibles al usuario final, pero que de forma inherente actúan sobre el resultado final de la aplicación y sobre la experiencia de desarrollo de la misma.

1.3.1. Funcionales

- Crear un conjunto de herramientas para la monitorización y el chequeo de diversos aspectos del estado de un servicio de Internet.
- Crear una aplicación online, de acceso público, que permita la creación y gestión de chequeos de manera sencilla, basada internamente en las herramientas mencionadas en el punto anterior.

¹Una página de *párking de dominio* sirve como página temporal mientras un servicio web no es lanzado, de forma que el dominio no esté en blanco.

- Habilitar a esta aplicación de un sistema de notificaciones mediante correo electrónico que alerte a los usuarios de posibles cambios en la disponibilidad de los servicios monitorizados.
- Crear una aplicación móvil para que los usuarios tengan la opción de recibir notificaciones instantáneas provenientes de la aplicación web con información de sus chequeos.

1.3.2. Transversales

- Investigar y conocer los vectores de vigilancia usados habitualmente para monitorizar servicios de Internet.
- Ampliar mis conocimientos sobre desarrollo web en general y las tecnologías de back-end en particular.
- Adquirir soltura en el uso del lenguaje de programación Python en entornos web.
- Obtener una base de conocimientos mínima sobre el desarrollo de aplicaciones sobre la plataforma móvil Android.
- Utilizar un enfoque de análisis, diseño y codificación orientado a objetos, de una forma lo más clara y modular posible, para permitir ampliaciones y modificaciones sobre la aplicación por terceras personas.
- Hacer uso de herramientas básicas en el desarrollo de software, como son los sistemas de control de versiones para llevar un control realista del desarrollo del software, así como hacer de las veces de sistema de copias de seguridad.

1.4. Estado del arte

Se detalla a continuación el estado del arte de los puntos de vigilancia de servicios más habituales y de los servicios web más similares a lo que se busca construir en el proyecto.

1.4.1. Métodos de monitorización

La monitorización de servicios en general, y de servicios remotos a través de Internet en particular, ha sido siempre un aspecto importante en el despliegue de cualquier proyecto. Existe un gran número de vectores de vigilancia que se han usado a lo largo de los años. A continuación se detallan los más habituales.

Envío de paquetes ICMP

ICMP (Internet Control Message Protocol) es uno de los principales protocolos de la familia TCP/IP. Es utilizado por dispositivos de red, como los enrutadores, para enviar mensajes de control con diversas funciones como, por ejemplo, mensajes de error informando de que una máquina remota no está disponible.

La herramienta más conocida que trabaja sobre ICMP es la utilidad **Ping**, que envía peticiones *eco* a un host remoto, y espera la respuesta. Durante ese proceso, la herramienta mide el tiempo desde el envío a la recepción, monitorizando las posibles pérdidas de paquetes y otros valores. Respuestas incorrectas o inexistentes permiten al usuario tener una idea de que la máquina remota no es accesible de forma correcta.

Un ejemplo de la salida del comando ping en un sistema GNU/Linux es la siguiente:

```
$ ping -c3 uca.es
PING uca.es (150.214.80.25) 56(84) bytes of data.
64 bytes from www21.uca.es (150.214.80.25): icmp_seq=1 ttl=48 time=43.3 ms
64 bytes from www21.uca.es (150.214.80.25): icmp_seq=2 ttl=49 time=42.7 ms
64 bytes from 150.214.80.25: icmp_seq=3 ttl=49 time=41.2 ms

--- uca.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 3207ms
rtt min/avg/max/mdev = 41.228/42.421/43.300/0.906 ms
```

El uso del comando ping para el chequeo básico de servidores es algo habitual desde hace más de dos décadas [48]. A pesar ello, hay muchos servidores que tienen deshabilitada su capacidad de responder a esta clase de paquetes, por lo que es imperativo contar con otras medidas de chequeo complementarias.

Chequeo de peticiones HTTP

A la hora de verificar el correcto funcionamiento de un servidor web, la opción más lógica es utilizar peticiones HTTP (Hyper Text Transfer Protocol) contra el servidor, tal y como haría un cliente web normal (como un navegador). El protocolo HTTP es la base de la comunicación en la web, y tiene sistemas para verificar que el envío y recepción de las peticiones es correcto.

Cuando un navegador u otro cliente hace una petición a un servidor web, éste devuelve una respuesta que comienza con un **código de estado**, seguido de unas cabeceras, una línea en blanco y el cuerpo del mensaje. El aspecto de las cabeceras es el siguiente

```
HTTP/1.1 200 OK
Server: nginx/1.4.1 (Ubuntu)
Date: Mon, 28 Apr 2014 17:53:26 GMT
```

Content-Type: text/html; charset=utf-8
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/5.5.3-1ubuntu2.2

En la primera línea se observa el **código de estado**, que indica el estado de la petición. Estos códigos fueron establecidos por la IETF (Internet Engineering Task Force), se componen de tres dígitos y contemplan un extenso número de situaciones, divididas en cinco familias según el primer dígito del código:

- **1xx, informativos.** Informan de que la petición ha sido recibida y que puede continuarse el proceso, por ejemplo enviando por parte del cliente el resto de la petición.
- **2xx, petición correcta.** Indica que la petición fue correctamente recibida, entendida y aceptada. El código más conocido de esta familia es el **200 - OK**, la respuesta estándar para peticiones correctas.
- **3xx, redirección.** Indica que se debe producir una redirección para completar la petición por parte del cliente. El código más popular en esta familia es el **302 - Moved temporarily**.
- **4xx, error del cliente.** Indica que el cliente ha cometido un error al realizar la petición, por ejemplo pidiendo un recurso inexistente. Es el caso del código **404 - Not Found**.
- **5xx, error del servidor.** Indica que aunque la petición parece correcta, el servidor ha sufrido un fallo al intentar responderla. El mensaje más habitual es el **500 - Internal Server Error**, que informa de manera genérica de un error interno.

Así, un punto de vigilancia habitual es comprobar que las peticiones hacia cierta URL (Uniform Resource Locator) devuelven el código de estado correcto. Que una web que normalmente devuelve un código 200 de pronto devuelva el código 500 es un indicador de que ha ocurrido un fallo y el sistema no funciona apropiadamente.

Incluso tras la verificación del correcto código de estado uno no puede estar del todo seguro de que una web funcione correctamente. Hay un caso especial de ataque, el *defacement* [50], que consiste en cambiar de forma total o parcial el contenido de una página web, manteniendo su accesibilidad. Así, las comprobaciones de tipo *ping* o de código de estado serán correctas, ya que el servidor seguirá en línea y respondiendo peticiones, pero el contenido no será el adecuado.

Para estos casos es importante establecer un sistema de verificación del contenido. La forma más sencilla y ligera es decidir una serie de palabras clave que deberán encontrarse en el contenido. Así, el sistema de vigilancia verificará en cada chequeo que esas palabras clave se encuentren en el cuerpo de la respuesta. Otra opción es generar inicialmente una imagen del contenido, ya sea haciendo una copia íntegra en la base de datos o guardando un resumen mediante un algoritmo de *hashing*, y compararla en cada chequeo. Con esto nos aseguramos de que

la totalidad del contenido coincide con lo esperado. La contrapartida es que este sistema no puede utilizarse en páginas dinámicas, ya que cualquier cambio en el contenido haría saltar la alerta.

Chequeo de puertos

El envío de paquetes ICMP está bien para comprobaciones genéricas, y el uso de peticiones HTTP es el más apropiado para servidores web, pero es necesario un término medio que sirva para vigilar otros servicios remotos de forma algo más exhaustiva pero manteniendo la adaptabilidad.

Estos servicios remotos utilizan **puertos lógicos** a través de los cuales los clientes pueden conectarse y obtener información. Los servidores web no son ninguna excepción, y suelen utilizar el puerto 80 como interfaz de comunicación con el exterior, pero hay muchos otros. Se pueden usar los números entre 0 y 65536 para identificar un puerto, y la IANA (Internet Assigned Numbers Authority) es responsable de asignar los números de puerto por debajo del 1024 a los servicios más habituales:

- 20 y 21: File Transfer Protocol (FTP)
- 22: Secure Shell (SSH)
- 23: Telnet remote login service
- 25: Simple Mail Transfer Protocol (SMTP)
- 53: Domain Name System (DNS)
- 80: Hypertext Transfer Protocol (HTTP)
- 110: Post Office Protocol (POP3)
- 119: Network News Transfer Protocol (NNTP)
- 143: Internet Message Access Protocol (IMAP)
- 161: Simple Network Management Protocol (SNMP)
- 194: Internet Relay Chat (IRC)
- 443: HTTP Secure (HTTPS)
- 465: SMTP Secure (SMTPS)

La verificación más habitual en este caso es abrir una conexión al puerto remoto, comprobando que la conexión se realiza correctamente y no es denegada por el servidor remoto. Opcionalmente se puede comprobar la respuesta textual del servidor, verificando que se obtiene un mensaje de bienvenida o algo similar, habitual en servicios como SSH o FTP. Por ejemplo, un servidor SSH puede devolver como línea inicial el siguiente mensaje:

```
SSH-2.0-OpenSSH_6.2p2 Ubuntu-6ubuntu0.1
```

Con esto nos cercioramos de que el servidor está en línea, es accesible, tiene abierto el puerto asociado al servicio, éste acepta conexiones y responde con el mensaje habitual.

Verificación de dominios y registros DNS

Hasta ahora, los mecanismos de monitorización que se han mencionado actúan directamente sobre el servidor remoto que ofrece el servicio a monitorizar, normalmente a través de una URL o un *hostname*. Desafortunadamente existen vectores de ataque que se dirigen hacia servicios relacionados pero no alojados o gestionados directamente por el servidor. Un ejemplo de esto es la gestión de los registros DNS (Domain Name Server).

En cualquier red, y en Internet en particular, los sistemas conectados se identifican generalmente utilizando una dirección IP (Internet Protocol), normalmente asignada por el ISP (Internet Service Provider) con el que se haya contratado el acceso o por el administrador de la red. Dado que recordar una dirección IP es algo difícil, surgen los **nombres de dominio**, cuyo principal propósito es el de ofrecer términos memorizables y fáciles de encontrar. Así, a un servidor con la ip 78.47.140.228 se le puede asignar el nombre de dominio `josetomastocino.com`, siendo ambos términos equivalentes.

El sistema DNS se encarga de gestionar y distribuir las asociaciones entre direcciones IP y nombres de dominio a través de los **registros DNS**. Para cada dominio es posible definir varios tipos de registros DNS, con diferente utilidad en cada caso. Los más habituales son los siguientes:

- Los registros de tipo **A** (*address record*) se utilizan para traducir nombres de servidores a direcciones IPv4. Es el registro más habitual.
- Los registros de tipo **AAAA** son similares a los tipo A pero para IPv6.
- Los registros **CNAME** (*canonical name*) se usan para crear nombres de servidor adicionales o *alias*. Es la herramienta a utilizar cuando se desea añadir un subdominio, como `subdominio.uca.es`.
- Los registros **MX** (*mail exchange*) sirven para asociar un dominio a una lista de servidores de correo, de forma que un agente de correo sepa a qué servidor enviar los mensajes.
- Los registros **TXT** (*Text*) guardan información textual, utilizada habitualmente como forma de identificación frente a servicios externos. Por ejemplo, en el caso de Google Apps, los dueños de los dominios deben incluir un registro TXT con un contenido similar a

`google-site-verification=rX0xyZounnZasA8Z7oaD3c14Jdj`

La información DNS asociada a un dominio se transmite a los clientes web a través de una estructura jerárquica de servidores DNS. Cuando se produce un cambio en

los registros DNS de un dominio, estos cambios no están disponibles inmediatamente a todos los usuarios, sino que se *propagan* de forma gradual a través todos los servidores de nombres del mundo. El proceso de propagación puede durar entre 10 y 72 horas, dependiendo de la zona geográfica de cada servidor.

El proceso de propagación de cambios, los errores en los servidores DNS y las actividades maliciosas de terceros pueden dar lugar a que, en un momento dado, un cliente no tenga la información DNS adecuada para un dominio. Puede ocurrir, por ejemplo, que a través de un ataque de *man-in-the-middle* [49] un usuario malicioso sirva a un cliente unos registros DNS incorrectos para un dominio en particular, redirigiendo a la víctima a otro servidor en el que puede producirse un robo de datos. Es por tanto importante establecer un sistema de vigilancia que permita comprobar que la información DNS asociada a un dominio sea la correcta en todo momento, revisando periódicamente la información de los registros.

1.4.2. Servicios similares

En la actualidad existen bastantes servicios web que ofrecen algunas características similares a las que se desean en SiteUp, pero no todas. Se presentan a continuación algunos de estos servicios, junto a las características que ofrecen y finalmente una comparativa con el proyecto que se presenta.

Pingdom <http://pingdom.com>

Pingdom es un servicio sueco de monitorización fundado en 2007. Tienen más de 400.000 usuarios en 200 países, con clientes tan importantes como Spotify, Instagram y Dropbox. Recientemente han actualizado su web con un nuevo diseño, y han añadido soporte para iOS y Android. Entre sus servicios se encuentran los clásicos chequeos por ping, por web, de protocolos y chequeos de interacciones de varios pasos.

Ofrecen una cuenta gratuita con la que probar el servicio que permite añadir un chequeo.

Alertra <http://alertra.com>

Alertra es un servicio de monitorización fundado en el año 2000 en Estados Unidos. Actualmente cuentan con bastantes tipos de chequeos, además de ofrecer información técnica sobre los chequeos que se realizan.

La cuenta gratuita está limitada a 30 días, tras los cuales hay que pagar.

Uptime Robot <http://uptimerobot.com>

Uptime Robot es una popular plataforma de monitorización desarrollada en Estados Unidos por un solo programador. Su principal atractivo es su cuenta gratuita,

que permite a los usuarios añadir hasta 50 chequeos con una periodicidad de 5 minutos sin coste alguno. Lamentablemente en los últimos meses su servicio ha ido teniendo problemas y su credibilidad ha caído, especialmente debido a no contar con un equipo completo de personal técnico detrás del producto.

StatusCake <http://statuscake.com>

StatusCake es uno de los principales servicios de monitorización de sitios web actuales. Se ofrece (literalmente) como una alternativa mejor que Uptime Robot, y cuenta con importantes clientes como Disney, Fender y Electronic Arts.

Su cuenta gratuita es una de las más generosas, ofreciendo chequeos ilimitados.

Tabla comparativa

En la figura 1.1 se hace una comparativa de los requisitos que se buscan en el proyecto, verificando si se ofrecen o no en las alternativas de servicio previamente estudiadas.

Una carencia importante bastante generalizada es la falta de aplicaciones nativas para móviles con las que recibir notificaciones sobre el estado de los servicios y gestionar los chequeos. Solo Pingdom y el presente proyecto, SiteUp, ofrecen aplicaciones para Android.

Así pues, queda patente la dificultad de encontrar un servicio que aúne el mayor número de características posibles a la vez que mantiene un servicio gratuito y de calidad, quedando manifiesta la necesidad del proyecto.

1.5. Alcance

SiteUp se modela como una herramienta de monitorización de servicios de Internet accesible a través de la web. Los usuarios tendrán la posibilidad de crear y gestionar una serie de *chequeos* de diversos tipos sobre los servicios web que elijan. La aplicación irá recopilando información relativa a esos chequeos, e informará al usuario en caso de que las verificaciones que se hayan dado de alta no coincidan con los resultados esperados.

Además, el usuario tendrá la posibilidad de recibir notificaciones de manera instantánea a través del correo electrónico y de una aplicación para la plataforma móvil **Android**. El servicio web estará totalmente adaptado para su uso en dispositivos móviles.

<i>Concepto</i>	Pingdom	Alertra	UptimeRobot	StatusCake	SiteUp
Chequeo mediante Ping	Sí	Sí	Sí	Sí	Sí
Chequeo de peticiones HTTP (código de estado)	No	No	No	Sí	Sí
Chequeo de peticiones HTTP (contenido)	Sí	Sí	Sí	Solo cuentas de pago	Sí
Chequeo de registros DNS	Limitado	No	No	Limitado	Sí
Chequeo de puertos	Sí	Sí	Sí	Sí	Sí
Chequeo de puertos (contenido)	Sí	Sí	No	No	Sí
Cuenta gratuita	Sí	30 días	Sí	Sí	Sí
Chequeos en cuenta gratuita	1	1	50	Ilimitado	Ilimitado
Frecuencia mínima en cuenta gratuita	1 minuto	5 minutos	5 minutos	5 minutos	1 minuto
Notificación por e-mail	Sí	Sí	Sí	Sí	Sí
Notificación por App Android	Sí	No	No	No	Sí
Open Source	No	No	No	No	Sí

Tabla 1.1: Tabla comparativa de servicios de monitorización

1.5.1. Licencia

El proyecto está publicado como software libre bajo la licencia GPL (General Public License) versión 3. El conjunto de bibliotecas y módulos utilizados tienen las siguientes licencias:

- El intérprete del lenguaje de programación **Python** se distribuye bajo la licencia PSFL (Python Software Foundation License), una licencia permisiva estilo BSD (Berkeley Software Distribution) y compatible con la GNU (GNU is Not Unix) GPL.
- **Django** [14], el framework de desarrollo web en el que se basa la aplicación, se distribuye bajo la licencia *BSD*.
- El servidor web **nginx** [32] está licenciado bajo la licencia BSD simplificada.
- Los siguientes paquetes de Python utilizan también la licencia BSD:
 - Celery
 - Sqlparse
 - iPython
 - dnspython
 - coverage
 - django-rest-framework
 - billiard
 - anyjson
 - Fabric
- Los siguientes paquetes de Python utilizan la licencia MIT (Massachusetts Institute of Technology):
 - PyDot
 - Gunicorn
 - Requests
 - django-extensions
 - six
 - sh
 - pip
 - virtualenvwrapper
 - factory-boy

1.6. Estructura del documento

El presente documento se rige según la siguiente estructura:

- **Introducción.** Se exponen el contexto, las motivaciones y objetivos detrás del proyecto **SiteUp**, así como información sobre las licencias de sus componentes, glosario y estructura del documento.
- **Planificación,** donde se explica la planificación del proyecto, la división de sus etapas, la extensión de las etapas a lo largo del tiempo y los porcentajes de esfuerzo.
- **Requisitos del sistema,** capítulo en el que se formalizan los objetivos y requisitos planteados en la introducción.
- **Análisis.** Se detalla la fase de análisis del sistema, explicando los requisitos funcionales del sistema, los diferentes casos de uso y bosquejando las interfaces visuales de los diferentes sistemas.
- **Diseño.** Tras el análisis, se expone en detalle la etapa de diseño del sistema, con los diagramas de las arquitecturas lógicas de los sistemas y los diagramas de diseño físico de datos.
- **Implementación.** Una vez analizado el sistema y definido su diseño, en esta parte se detallan las decisiones de implementación más relevantes que tuvieron lugar durante el desarrollo del proyecto.
- **Pruebas.** Se describen las pruebas que se han llevado a cabo sobre el proyecto para garantizar su fiabilidad y consistencia.
- **Conclusiones.** Comentamos las conclusiones a las que se han llegado durante el transcurso y al término del proyecto.

Y los siguientes apéndices:

- **Manual de instalación de la plataforma web** en sistemas nuevos.
- **Manual de usuario,** donde se explica cómo usar la aplicación.

1.7. Acrónimos

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

Sass Syntactically Awesome Style Sheets

AJAX Asynchronous JavaScript and XML

SQL Structured Query Language

ORM Object-relational mapping

AMQP Advanced Message Queuing Protocol

LOPD Ley Orgánica de Protección de Datos

WSGI Web Server Gateway Interface

PSFL Python Software Foundation License

API Application Programming Interface

BSD Berkeley Software Distribution

CRUD Create-Read-Update-Delete

CSS Cascading Style Sheet(s)

DNS Domain Name Server

FDL Free Documentation License

FSF Free Software Foundation

GNU GNU is Not Unix

GPL General Public License

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

ICMP Internet Control Message Protocol

ISP Internet Service Provider

IP Internet Protocol

MIT Massachusetts Institute of Technology

MVC Model View Controller

PDF Portable Document Format

SVG Structured Vector Graphics

URL Uniform Resource Locator

VPS Virtual Private Server

Capítulo 2

Planificación

En este capítulo procederemos a detallar la planificación que ha seguido el proyecto. En particular, se describe la metodología seguida, las etapas por las que ha pasado el proyecto, la planificación temporal y la organización del personal involucrado.

Tras ello se exponen los recursos inventariables de los que se han hecho uso a lo largo del desarrollo y puesta en producción del producto, haciendo una estimación de los costes y, por último, explicando los posibles riesgos que pueden tener lugar.

En general, el desarrollo del proyecto se ha ajustado razonablemente bien al calendario inicialmente dispuesto en la planificación, con algunas etapas durando más de lo previsto pero otras terminando antes de lo planificado.

2.1. Metodología de desarrollo

Para la realización del proyecto se ha utilizado un modelo de desarrollo iterativo incremental, definiendo en cada etapa las funcionalidades a elaborar y completando el ciclo completo de desarrollo y pruebas.

A continuación se detallan cada una de las iteraciones por las que ha ido pasando el proyecto.

2.1.1. Primera iteración: adquisición de conocimientos y elicitación de requisitos

Antes de poder comenzar con el análisis y diseño del propio proyecto, era esencial adquirir una serie de conocimientos para poder afrontar su desarrollo con todas las garantías. Durante esta iteración, se llevaron a cabo labores de documentación y aprendizaje autodidacta con las que se asentaron los conocimientos necesarios.

Además, durante este periodo también se definieron los requisitos que se impondrían al sistema, y se barajaron las diferentes posibilidades de implementación del proyecto, así como las posibles herramientas y bibliotecas de terceros que pudieran ser de ayuda, tal y como queda reflejado en el capítulo 3.

En particular, en esta etapa se decidió el uso del framework web **Django**, las tecnologías de desarrollo *front-end*¹ (entre otras, Sass, Compass, jQuery y D3), el motor de tareas asíncronas para lanzar los chequeos (Celery y RabbitMQ) y el stack de soporte del servidor (nginx, supervisord y gunicorn).

2.1.2. Segunda iteración: desarrollo de módulo de herramientas básicas de chequeo

Una vez adquiridos los conocimientos necesarios, y decididas las técnicas y herramientas para llevar aquellos a la práctica, fue obvia la necesidad de empezar por diseñar una serie de herramientas que fuesen capaces de lanzar chequeos contra servicios web de forma simple y aislada.

Así, se desarrolló un módulo en Python capaz de lanzar los cuatro tipos de chequeos con los que posteriormente contaría el proyecto:

- Chequeo mediante paquetes ICMP (ping) con verificación de tiempo de espera máximo.
- Chequeo de coherencia de registros DNS.
- Chequeo del estado de puertos remotos.
- Chequeo de peticiones HTTP, tanto en su cabecera como en contenido.

Este módulo sería el motor de los chequeos que posteriormente se darían de alta en el proyecto.

2.1.3. Tercera iteración: inicio de proyecto Django y CRUD básico

Con el módulo de chequeo desarrollado, *sólo* restaba desarrollar el resto de la aplicación alrededor de sus funcionalidades. En esta tercera iteración se creó la estructura básica del proyecto y se inició el desarrollo de la funcionalidad CRUD (Create-Read-Update-Delete) básica.

También en esta etapa se definió el diseño visual de la aplicación: logotipo, esquema de colores y tipografías.

¹El desarrollo web *front-end* hace referencia a la parte de una web con la que el usuario interactúa directamente en el navegador.

2.1.4. Cuarta iteración: integración del motor de tareas asíncronas

Con la aplicación teniendo la funcionalidad básica para la creación y edición de chequeos, el siguiente paso fue integrar un motor de tareas asíncronas que se dedicase a revisar y lanzar los chequeos dados de alta en el sistema, guardando el resultado de cada uno de ellos en la base de datos y generando estadísticas.

2.1.5. Quinta iteración: desarrollo de la aplicación Android

Tras concluir el desarrollo de la aplicación web, en esta etapa se desarrolló una aplicación móvil para el sistema operativo Android que recibe notificaciones con información sobre los resultados de los chequeos dados de alta en el sistema.

2.2. Planificación temporal

Se ha diseñado un diagrama de Gantt para reflejar la distribución de las tareas a lo largo del tiempo (figura 2.1, en la página 40).

Además, se presenta a continuación una comparación por fases de los plazos estimados frente a los plazos reales tras la conclusión del proyecto.

Fase	Plazo estimado	Plazo real
Planificación	4 días	5 días
Aprendizaje preeliminar	15 días	13 días
Módulo de chequeo	10 días	14 días
Plataforma web	30 días	38 días
Motor de tareas	14 días	12 días
Aplicación Android	21 días	19 días
Edición de la documentación	28 días	18 días

Tabla 2.1: Comparación de la estimación con los tiempos reales

2.3. Organización

El proyecto ha sido desarrollado en su totalidad por el que escribe, alumno de la Universidad de Cádiz, llevando a cabo las labores de análisis, diseño y desarrollo, así como de diseño visual de las interfaces y *branding* del proyecto. El desarrollo del proyecto ha sido revisado y guiado de forma continua por el tutor Iván Ruiz Rube.

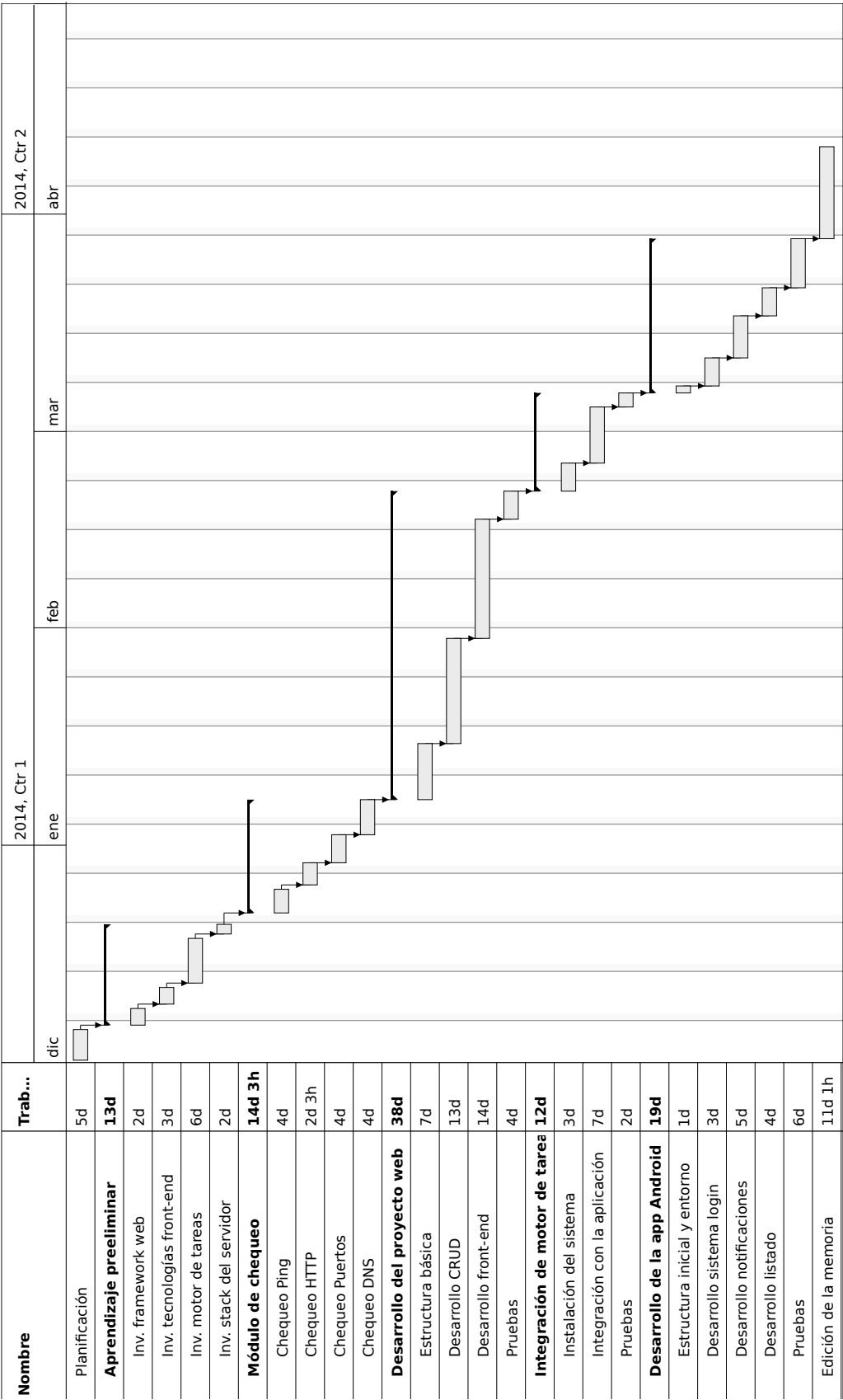


Figura 2.1: Diagrama de Gantt

2.4. Recursos inventariables

Los recursos de hardware utilizados durante el desarrollo y la implantación del proyecto engloban aquellos empleados por el alumno para la elaboración del proyecto y los necesarios para la implantación y puesta en marcha del proyecto.

En particular:

- Como puesto de desarrollo se ha utilizado un equipo con las siguientes características:
 - Procesador Intel i5 4670k
 - Placa Gigabyte Z87X-OC
 - Memoria Corsair Vengeance 16GB DDR3
 - Gráfica NVidia GeForce GTX 660 Ti
 - SSD Crucial M4 128GB
 - Pantalla Dell u2713H 27"
- Como servidores para el *deployment* se han contratado los servicios de la empresa alemana Hetzner, haciendo uso del plan *vServer VQ7* que proporciona un servidor virtual con las siguientes características:
 - Procesador Single Core
 - RAM 512MB
 - Disco Duro 20GB
 - NIC 100Mbit
 - Tráfico máximo 1TB al mes
- Como dispositivo para las pruebas de la aplicación móvil se ha utilizado un **Samsung Galaxy Nexus** con las siguientes características:
 - Sistema: Android 4.3
 - CPU: 1.2 GHz ARM Cortex-A9 de doble núcleo.
 - GPU: 307 MHz PowerVR SGX540.
 - Memoria: 1 GB.
 - Almacenamiento: 16GB
 - Pantalla: 4.65"HD Super AMOLED, 1280x720px.

2.5. Costes

La estimación de los costes del proyecto es divisible en dos áreas. Por un lado, en cuanto al personal:

- Se requiere de un desarrollador que haga el análisis, diseño e implementación de la plataforma web y la aplicación Android. Según convenio [6], en la categoría de *Programador en Internet*, el sueldo base es de 17€/hora.
- Se precisa de un administrador del sistemas cualificado que lleve el mantenimiento periódico del servidor de despliegue. Según convenio, el sueldo base es de 14€/hora.

Dado el contexto del proyecto, es el alumno el que cubre ambos perfiles. Por otro lado, la infraestructura necesaria consta de:

- El sistema en el que se ha llevado a cabo el desarrollo. Estaba comprado con anterioridad al proyecto, por lo que no se ha reflejado en este despliegue de gastos.
- La plataforma de despliegue del sistema. Se trata de un VPS (Virtual Private Server) alquilado mensualmente con un coste fijo de 7.90€ al mes [24], impuestos incluidos.

Así, haciendo uso de lo dispuesto en la sección 2.2, *Planificación temporal* y suponiendo un periodo mínimo de actividad de 12 meses, se elabora la tabla de planificación de gastos del cuadro 2.2.

Concepto	Coste unitario	Unidades	Coste total
<i>Desarrollador</i>	17€/hora	480 horas	8160€
<i>Administrador</i>	14€/hora	48 horas	672€
<i>Servidor VPS</i>	7.90€/mes	12 meses	94.80€
Total			8926.80€

Tabla 2.2: Tabla de planificación de costes

2.6. Riesgos

Los principales riesgos presentes en SiteUp son los siguientes.

2.6.1. Compromiso de la plataforma de despliegue

Dada la limitada envergadura del proyecto por ser éste un PFC y no un desarrollo con fines comerciales, los chequeos de los servicios web se hacen de forma centralizada desde la plataforma de despliegue. Cualquier problema que pueda comprometer la estabilidad de esta plataforma supone un riesgo de obtener datos

falsos de los chequeos. En un entorno real, para obtener la mayor fiabilidad y datos siempre precisos, cada chequeo se lanza desde diversos puntos, tanto geográficos como lógicos, de forma que es más difícil que un falso positivo o negativo causado por un error de la plataforma pueda llegar hasta el usuario.

Lógicamente la adición de nuevas *sondas* de chequeo incrementaría enormemente la complejidad logística y el presupuesto del proyecto.

2.6.2. Limitación de recursos

SiteUp hace uso de un gran número de sistemas, desde el servidor web que recibe las peticiones hasta la cola de tareas asíncronas. La plataforma de despliegue cuenta con unos recursos limitados que, en situaciones de gran afluencia de peticiones, pueden no ser suficientes y dar lugar a problemas en el sistema.

Este riesgo puede superarse considerando contratar planes con más recursos para la plataforma, lo que por otro lado conllevaría un incremento en los gastos.

2.6.3. Rechazo de los servicios vigilados

Por definición, los sistemas de vigilancia tienen que lanzar chequeos de forma periódica y repetida a lo largo del tiempo. Algunos sistemas están configurados para detectar esta clase de comprobaciones y bloquearlas, al identificarlas (a veces erróneamente) como ataques de denegación de servicio [52].

Este riesgo es algo que tenemos que asumir y frente al que no se puede hacer mucho, aparte de configurar la periodicidad de las comprobaciones para disminuir la frecuencia y que no se tomen como un ataque.

Capítulo 3

Requisitos del sistema

En este capítulo procederemos a describir de manera formal los objetivos que se presentaron de manera genérica en el apartado 1.3, *Objetivos*.

En particular, se detallan:

- Los **objetivos del sistema**, que resumen a muy grandes rasgos la funcionalidad del proyecto.
- Los **requisitos funcionales**, que definen las funciones del sistema software y sus componentes.
- Los **requisitos de información**, que detallan los datos que el sistema usará durante el desarrollo y ejecución para llevar a cabo de forma adecuada su funcionalidad.
- Los **requisitos no funcionales**, que, en general, especifican criterios de diversas categorías que el software debe cumplir para garantizar un buen nivel de calidad más allá de su funcionamiento.

Para terminar el capítulo se presentan diversas alternativas tecnológicas para cubrir las necesidades surgidas en los requerimientos del sistema, detallando las decisiones tomadas.

3.1. Objetivos del sistema

Los objetivos principales de SiteUp, definidos a alto nivel son los dos siguientes:

Título	Crear chequeos de varios tipos sobre servicios de Internet
Descripción	SiteUp permitirá crear chequeos de diversas clases para la comprobación de servicios de Internet, utilizando varias tecnologías de chequeo y múltiples vectores de verificación: ping, puertos, hipertexto y registros DNS.

Tabla 3.1: OBJ-1, objetivo del sistema 1

Título	Notificación a los usuarios
Descripción	SiteUp deberá notificar a sus usuarios cuando el estado de sus chequeos cambie. Estas notificaciones se harán tanto por móvil como a través de una aplicación Android.

Tabla 3.2: OBJ-2, objetivo del sistema 2

3.2. Catálogo de requisitos

Se presentan a continuación los requisitos del sistema, tanto a nivel funcional como de información y no funcionales.

3.2.1. Requisitos funcionales

En esta subsección se introducen los requisitos funcionales, que describen las funciones del sistema software.

Gestión de usuarios

Título	Creación de cuenta de usuario
Descripción	El usuario de SiteUp deberá ser capaz de crear una cuenta de usuario para acceder a las funciones de SiteUp, proporcionando su nombre de usuario, dirección de correo electrónico y contraseña.

Tabla 3.3: RQF-1, requisito funcional 1: creación de cuenta de usuario

Título	Edición de cuenta de usuario
Descripción	Un usuario logueado deberá ser capaz de editar sus datos personales y preferencias de usuario. En particular, deberá ser capaz de editar su nombre de usuario, dirección de correo electrónico y contraseña, además de opciones como la posibilidad de recibir un resumen diario del estado de sus chequeos.

Tabla 3.4: RQF-2, requisito funcional 2: edición de cuenta de usuario

Gestión de grupos de chequeos

Título	Creación de grupos de chequeos
Descripción	Un usuario logueado deberá ser capaz de crear un grupo de chequeos, indicando el título que identifique al grupo.

Tabla 3.5: RQF-3, req. funcional 3: creación de grupo de chequeos

Título	Edición de grupos de chequeos
Descripción	Un usuario logueado deberá ser capaz de editar un grupo de chequeos que previamente haya creado. En particular, deberá ser capaz de modificar el título del grupo.

Tabla 3.6: RQF-4, req. funcional 4: edición de grupo de chequeos

Título	Eliminación de grupos de chequeos
Descripción	Un usuario logueado deberá ser capaz de eliminar un grupo de chequeos, eliminando en el proceso tanto el grupo en sí como los chequeos que pertenezcan a ese grupo.

Tabla 3.7: RQF-5, req. funcional 5: eliminación de grupo de chequeos

Título	Activación y desactivación de grupos de chequeos
Descripción	Un usuario logueado deberá ser capaz de activar y desactivar un grupo de chequeos. En la práctica, esto activará o desactivará cada uno de los chequeos que pertenezcan al grupo de forma individual.

Tabla 3.8: RQF-6, req. funcional 6: activación de grupo de chequeos

Gestión de chequeos

Título	Creación de chequeos
Descripción	Un usuario logueado deberá ser capaz de añadir a un grupo un chequeo que puede ser de cuatro tipos distintos: <i>Ping check</i> , <i>Port check</i> , <i>DNS Check</i> y <i>HTTP Check</i> . Según el tipo de chequeo a dar de alta, el usuario deberá introducir una serie de detalles, siendo común a todos ellos el incluir un título que identifique el chequeo, el objetivo del chequeo, la periodicidad y las opciones de notificar mediante e-mail y Android. Los detalles particulares de cada chequeo se definen en los requisitos de información.

Tabla 3.9: RQF-7, req. funcional 7: creación de chequeos

Título	Actualización de chequeos
Descripción	Un usuario logueado deberá ser capaz de actualizar un determinado chequeo, modificando cualquiera de los detalles que lo definen (a excepción del tipo de chequeo, que es fijo).

Tabla 3.10: RQF-8, req. funcional 8: actualización de chequeos

Título	Eliminación de chequeos
Descripción	Un usuario logueado deberá ser capaz de eliminar un determinado chequeo, evitando así que el sistema lo tenga en cuenta a la hora de lanzar las monitorizaciones.

Tabla 3.11: RQF-9, req. funcional 9: eliminación de chequeos

Título	Activación y desactivación de chequeos
Descripción	Un usuario logueado deberá ser capaz de activar o desactivar un chequeo, de forma que mientras esté activado, el chequeo será lanzado periódicamente por el motor de chequeos, y cuando esté desactivado el chequeo será ignorado por el sistema hasta que el usuario lo active de nuevo.

Tabla 3.12: RQF-11, req. funcional 11: activación de chequeos

Título	Revisión de chequeos
Descripción	Un usuario logueado deberá ser capaz de revisar los datos de monitorización que se hayan generado para un chequeo que previamente haya dado de alta, pudiendo ver una gráfica con el estado del chequeo a lo largo del tiempo. Además debe ser capaz de elegir el período de tiempo en el que revisar los datos (últimas 24 horas, última semana, etcétera).

Tabla 3.13: RQF-11, req. funcional 11: revisión de chequeos

Título	Recepción de notificaciones por correo electrónico
Descripción	Un usuario logueado deberá ser capaz de recibir notificaciones a través del correo electrónico cuando el estado de un chequeo cambie, siempre que el usuario haya activado las notificaciones por correo para ese chequeo.

Tabla 3.14: RQF-12, req. funcional 12: notificación por correo

Título	Lanzamiento de chequeos
Descripción	El sistema deberá ser capaz de obtener los chequeos dados de alta y lanzar los procesos pertinentes para realizar las verificaciones que esos chequeos necesiten, generando unos registros o <i>logs</i> con la información obtenida en cada momento.

Tabla 3.15: RQF-13, req. funcional 13: lanzamiento de chequeos

Título	Mantenimiento de registros de chequeo
Descripción	El sistema deberá ser capaz de realizar de forma periódica un mantenimiento de los registros de chequeo (<i>logs</i>). En particular, deberá llevar hacer un <i>resumen</i> de registros de forma que se reduzca el número de elementos en la base de datos manteniendo a la vez el grueso de la información.

Tabla 3.16: RQF-14, req. funcional 14: mantenimiento de registros

Aplicación Android

Título	Listado de chequeos en la aplicación Android
Descripción	Un usuario deberá ser capaz de ejecutar la aplicación de Android, loguearse con su cuenta de usuario y ver un listado de los chequeos que ha dado de alta en el sistema, así como un resumen de su estado.

Tabla 3.17: RQF-15, req. funcional 15: listado de chequeos en app

Título	Recepción de notificaciones por la aplicación Android
Descripción	Un usuario logueado deberá ser capaz de recibir notificaciones a través de la aplicación Android cuando el estado de un chequeo cambie, siempre que el usuario haya activado las notificaciones para ese chequeo.

Tabla 3.18: RQF-16, req. funcional 16: notificaciones mediante la app

3.2.2. Requisitos de información

A continuación se presentan los requisitos de información, que describen los datos necesarios que requiere la aplicación para operar.

Título	Usuario del sistema
Datos específicos	<ul style="list-style-type: none"> ■ Nombre de usuario ■ Dirección de correo electrónico ■ Contraseña ■ (Opcional) Identificador de dispositivo Android

Tabla 3.19: IRQ-1, req. de información 1: usuario

Título	Chequeo tipo Ping
Datos específicos	<ul style="list-style-type: none"> ■ Título ■ Descripción ■ Objetivo – Debe ser una IP o un hostname ■ Frecuencia de chequeo en minutos ■ Notificar por correo ■ Notificar por Android ■ Si se debe chequear el tiempo de respuesta ■ Tiempo máximo de respuesta en milisegundos

Tabla 3.20: IRQ-2, req. de información 2: chequeo Ping

Título	Chequeo tipo DNS
Datos específicos	<ul style="list-style-type: none"> ■ Título ■ Descripción ■ Objetivo – Debe ser un hostname ■ Tipo de registro DNS (A, AAAA, CNAME, MX, TXT) ■ Contenido esperado del registro DNS ■ Frecuencia de chequeo en minutos ■ Notificar por correo ■ Notificar por Android

Tabla 3.21: IRQ-3, req. de información 3: chequeo DNS

Título	Chequeo tipo Port
Datos específicos	<ul style="list-style-type: none"> ■ Título ■ Descripción ■ Objetivo – Debe ser una IP o un hostname ■ Puerto objetivo ■ <i>Opcional</i> - Cadena de caracteres que debe aparecer en la respuesta ■ Frecuencia de chequeo en minutos ■ Notificar por correo ■ Notificar por Android

Tabla 3.22: IRQ-4, req. de información 4: chequeo de puerto

Título	Chequeo tipo HTTP
Datos específicos	<ul style="list-style-type: none"> ■ Título ■ Descripción ■ Objetivo – Debe ser una URL HTTP ■ Código de estado HTTP ■ <i>Opcional</i> - Cadena de caracteres que debe aparecer en la respuesta ■ Frecuencia de chequeo en minutos ■ Notificar por correo ■ Notificar por Android

Tabla 3.23: IRQ-5, req. de información 5: chequeo HTTP

Título	Registro de un chequeo
Datos específicos	<ul style="list-style-type: none"> ■ Fecha y hora de registro ■ Estado (Up, Down, Error) ■ Tiempo de respuesta (solo para chequeos tipo Ping). ■ Información adicional ■ Chequeo asociado

Tabla 3.24: IRQ-6, req. de información 6: registro de chequeo

Título	Estado de un chequeo
Datos específicos	<ul style="list-style-type: none"> ■ Fecha y hora de inicio del estado ■ Fecha y hora de fin del estado ■ Estado (Up, Down, Error) ■ Información adicional ■ Chequeo asociado

Tabla 3.25: IRQ-7, req. de información 7: estado de chequeo

Título	Grupo de chequeos
Datos específicos	<ul style="list-style-type: none"> ■ Título ■ Dueño

Tabla 3.26: IRQ-8, req. de información 8: grupo de chequeos

3.2.3. Requisitos no funcionales

Seguidamente se presentan los requisitos no funcionales, que especifican criterios usados para juzgar la operación del software a través de varias categorías, más allá de su comportamiento específico.

Requisitos de seguridad

Título	Cifrado de contraseñas de usuario
Descripción	Las contraseñas de los usuarios se almacenarán cifradas con un algoritmo de un solo sentido, de acuerdo a lo establecido en la vigente LOPD.

Tabla 3.27: NRQ-1, req. no funcional 1: cifrado de contraseñas

Título	Restricciones de acceso a usuarios
Descripción	Los datos de los chequeos dados de alta sólo serán accesibles por los usuarios que los hayan creado y los administradores de la plataforma.

Tabla 3.28: NRQ-2, req. no funcional 2: control de acceso

Requisitos de fiabilidad

Título	Ejecución de chequeos
Descripción	Los retrasos en la periodicidad de los chequeos no puede superar el 200 % del tiempo entre chequeos. Esto es, si un chequeo tiene una periodicidad de 1 minuto, en caso de sobrecarga del sistema el tiempo máximo entre chequeos será de 3 minutos (un retraso del 200 %).

Tabla 3.29: NRQ-3, req. no funcional 3: ejecución de chequeos

Requisitos de accesibilidad y usabilidad

Título	Adaptación a dispositivos
Descripción	La plataforma web deberá ser accesible desde cualquier clase de dispositivo móvil con acceso a Internet: móviles, tablets, etcétera, manteniendo siempre la accesibilidad y el grueso del contenido.

Tabla 3.30: NRQ-4, req. no funcional 4: adaptación a dispositivos

Requisitos de eficiencia

Título	Monitorización de peticiones de demasiada duración
Descripción	En aras de mantener el sistema fluyendo, el motor de chequeos deberá ser capaz de identificar y abortar los chequeos que, por causas propias o ajenas al sistema, estén tardando más de lo previsto e incidiendo negativamente en la eficiencia del sistema.

Tabla 3.31: NRQ-5, req. no funcional 5: monitorización de la ejecución

3.3. Alternativas de solución

En esta sección, se presentan diferentes alternativas tecnológicas que permiten satisfacer los requerimientos del sistema, presentando las alternativas elegidas y detallando las razones de esta decisión.

3.3.1. Frameworks para el desarrollo de la plataforma web

Dada la mayor familiaridad del desarrollador con el lenguaje, solo se barajaron frameworks de desarrollo basados en el lenguaje Python.

- **Django**[14]: popular framework para el desarrollo rápido de aplicaciones web. Consta de una arquitectura similar al popular patrón MVC (Model View Controller), y cuenta con una vasta comunidad de usuarios. Los proyectos Django se organizan en *apps*, que permiten modularizar la funcionalidad. Integra un ORM (Object-relational mapping), un sistema de plantillas y una sección de administración adaptable a cualquier sitio web.
- **Flask**[18]: se trata de un minimalista framework para Python que provee las funcionalidades más básicas, dejando al desarrollador elegir el resto de componentes. En los últimos meses, Flask ha ganado gran popularidad frente a Django al dar más libertad al programador.
- **Bottle**[7]: similar a Flask, Bottle es otro micro-framework para Python, siendo tan minimalista que está encapsulado en un solo fichero fuente y no necesita de dependencias externas. Se encuentra en etapas tempranas de su desarrollo.

Se tomó la decisión de optar por **Django** por ser el framework más robusto y con mayor número de utilidades. Su uso por grandes proyectos, como Instagram, afianzó aún más la decisión de usarlo.

3.3.2. Sistemas de gestión de bases de datos

La plataforma web contará con un modelo de información que se habrá de soportar mediante un sistema de gestión de bases de datos. Se presentan a continuación las alternativas barajadas.

- **MySQL**[31], uno de los sistemas de bases de datos más populares. Se usa en multitud de proyectos de gran envergadura y su utilidad y fiabilidad están más que demostradas. Cuenta con una arquitectura cliente-servidor, y suele necesitar bastantes recursos.
- **SQLite**[40]: potente pero ligero, SQLite se aleja de la arquitectura cliente-servidor y se integra directamente en la aplicación cliente pasando a ser parte integral del mismo. Esto reduce la latencia en las comunicaciones de datos. Es el SGBD que Django usa por defecto.

En este aspecto, se ha tomado la decisión de utilizar SQLite por varias razones. En primer lugar, es la opción que usa Django por defecto. En segundo lugar, los datos se guardan en un fichero, por lo que resulta trivial mover el sistema de un servidor a otro, lo cual es un gran ahorro de tiempo durante el desarrollo. Finalmente, SQLite provee el rendimiento necesario para las etapas tempranas del proyecto.

De cualquier modo, en un futuro no se descarta pasar a un SGBD basado en la arquitectura cliente-servidor, como MySQL.

3.3.3. Sistemas para la ejecución asíncrona de tareas

Dada la naturaleza de la *vigilancia* es indispensable contar con algún sistema de ejecución periódica y asíncrona de tareas. Se presentan a continuación las alternativas más habituales.

- **Cron**[51] es un demonio ¹ encargado de lanzar procesos de forma periódica utilizado en sistemas operativos basados en Unix. Cada usuario en un sistema cuenta con un fichero *crontab*, en el que puede definir qué tareas ejecutar y con qué periodicidad deben ejecutarse.
- **Celery**[9] es una cola de tareas asíncronas desarrollado en Python. Una instalación de celery cuenta con una serie de *workers* que leen tareas de una cola de mensajes (basada en el estándar AMQP (Advanced Message Queuing Protocol)) y las ejecutan de forma síncrona o asíncrona, opcionalmente guardando su resultado en alguna clase de estructura.

De las opciones presentadas finalmente se decidió usar **Celery** por varias razones:

- Facilidad al dar de alta nuevas tareas. En el caso de cron, es necesario gestionar manualmente las tareas a ejecutar y añadir también de forma manual, el proceso que lance esas tareas al fichero *crontab*. En el caso de celery, contamos con unas bibliotecas con las que simplemente damos de alta una tarea y el sistema se encarga de ponerla en cola y ejecutarla en el momento adecuado.
- Monitorización. Una de las mayores dificultades que presenta cron es la monitorización de su ejecución. No existe una certeza absoluta de que se estén lanzando los procesos, al menos que se elabore un sistema de verificación externo. En cambio, celery cuenta con sistemas de monitorización de la ejecución y *logs* en los que se registra el progreso de cada tarea.
- Integración. Al estar escrito en Python, celery se integra perfectamente con cualquier proyecto Django. Sin embargo, cron no ofrece de forma nativa ninguna opción de integración.

3.3.4. Plataformas para el desarrollo de la aplicación móvil

Como se especificó en la sección 1.3, uno de los objetivos principales del proyecto era el desarrollo, junto a la plataforma web, de una aplicación móvil para que los usuarios pudiesen recibir notificaciones instantáneas en cualquier momento y lugar, directamente en su móvil. Así pues, resultó necesario hacer un estudio de las plataformas móviles existentes para ver en cuál de ellas se centraría el desarrollo.

¹Entiéndase *demonio* en el sentido de un servicio que se ejecuta en segundo plano.

- **iOS**[5] es un sistema operativo móvil de la empresa **Apple** utilizado en los dispositivos de la familia iPhone, iPod Touch e iPad. A pesar de estar basado en un núcleo BSD libre, iOS tiene una licencia privativa.

El desarrollo para iOS es bastante estricto. Sólo es posible desarrollar para iOS utilizando computadoras Mac con el sistema operativo OS X 10.8 o superior[41]. La programación se lleva a cabo utilizando **Objective-C**, un lenguaje derivado del C cuyo uso a día de hoy prácticamente se limita al desarrollo en iOS. Además, es necesario utilizar el conjunto de herramientas que provee Apple, en particular XCode.

El control de Apple sobre las aplicaciones que se desarrollan es férreo. La *App Store*, aplicación desde la que los usuarios instalan nuevos programas, controla fuertemente las aplicaciones que se envían mediante exhaustivas revisiones, que suelen tardar entorno a una semana. Además, para poder lanzar una app es necesario pagar una cuota anual de 99 dólares.

- **Android**[3] es un sistema operativo móvil basado en el kernel de Linux, diseñado para su uso en móviles y tabletas. Inicialmente su desarrollo fue llevado a cabo por la empresa Android, Inc, pero posteriormente Google compró la empresa y es la que en la actualidad impulsa el desarrollo.

El desarrollo en Android es bastante más laxo que en iOS. El lenguaje de programación utilizado es Java, un lenguaje compilado muy conocido y utilizado en muchos ámbitos, más allá del desarrollo móvil. Es posible desarrollar en cualquier plataforma (GNU/Linux, Windows o Mac), utilizando cualquier entorno de desarrollo que sea capaz de compilar código Java con las bibliotecas de Android.

Las aplicaciones desarrolladas para Android se pueden lanzar en un mercado de aplicaciones llamado *Google Play Store*, aunque también existen mercados de terceros al igual que es posible instalar aplicaciones de forma manual. Google cobra una tarifa única de 25 dólares por desarrollador, válida de por vida. Las revisiones que hace Google de las aplicaciones son menos rígidas que en el caso de Apple, y suelen tardar mucho menos.

- **BlackBerry, Windows Phone, Symbian y Firefox OS** son otras plataformas móviles de mucho menor calado y público. El desarrollo para esas plataformas está mucho menos extendido, tiene menor soporte y menor clientela.

BlackBerry OS es un sistema operativo que nació en 1999, época en la que BlackBerry era la compañía estrella en dispositivos *handheld*, siendo la opción principal para empresarios y gente de negocios que tuviese necesidad de chequear el correo electrónico o notas estando lejos de una computadora. En la actualidad, el número de dispositivos que utilizan BlackBerry ha caído estrepitosamente, principalmente por la limitada compatibilidad del sistema BlackBerry 10 con otros dispositivos.

Windows Phone es la apuesta de Microsoft en el mercado móvil. Lanzado en 2010, la última versión del sistema es Windows Phone 8.1. Actualmente

se encuentra en pleno crecimiento, principalmente gracias al dominio de Nokia en mercados crecientes como Asia, América Latina y África[46]. A pesar de ello, todavía no es una plataforma firme con una base de usuarios a considerar.

Symbian es una plataforma libre que tuvo un gran éxito en la década de 2000, siendo la alternativa más asequible a BlackBerry hasta la aparición de Android en 2010. La última versión del sistema se lanzó en octubre de 2012, y desde entonces no habido más desarrollo.

Firefox OS es uno de los últimos sistemas en llegar, liderado por Mozilla y desarrollado entorno a tecnologías como HTML5. Su despegue está siendo lento y las críticas no son buenas, sobre todo por la falta de funcionalidades y errores. Además, el elenco de dispositivos compatibles todavía es muy pequeño y de bajas prestaciones.

Cuotas de mercado

En la tabla 3.32 se detallan las cuotas de mercado de las principales plataformas móviles medidas en enero de 2014 entre usuarios mayores de 13 años de edad [11].

<i>Plataforma</i>	<i>Porcentaje</i>
Android	51.7 %
Apple	41.6 %
BlackBerry	3.1 %
Microsoft	3.2 %
Symbian	0.2 %

Tabla 3.32: Cuota de mercado de plataformas móviles en enero de 2014

Decisión

Así, dadas las evidencias en la cuota de mercado, facilidad de desarrollo y costes en licencias, se tomó la decisión de usar **Android** como plataforma móvil principal a la hora de desarrollar la aplicación móvil.

En un futuro, según las circunstancias, se valorará trasladar el desarrollo también a la plataforma iOS, aunque es algo harto improbable dadas las exigencias de infraestructura (es necesario tener un ordenador Mac reciente) y monetarias (el coste de la licencia anual es prohibitivo para una aplicación gratuita).

Capítulo 4

Análisis

En este capítulo se dispone el análisis realizado en base a los requisitos presentados en el capítulo 3, haciendo uso del lenguaje de modelado UML. Concretamente, el capítulo describe:

- El **modelo conceptual** de los datos usados en la aplicación, elaborado a partir de los requisitos de información.
- Los **casos de uso**, que describen los pasos que componen los procesos habituales del proyecto.
- El **modelo de interfaz de usuario**, en el que se presenta un prototipo de la navegación y de las interfaces.

4.1. Modelo conceptual

De acuerdo a lo reflejado en el apartado 3.2.2 se presenta el diagrama 4.1 en el que se identifican los principales tipos de datos, sus atributos y las relaciones entre ellos.

A continuación se detallan cada uno de los tipos de datos reflejados en el modelo conceptual.

4.1.1. Usuario

Representa a un usuario de la aplicación que tiene intención de dar de alta cheques.

Nombre Nombre elegido por el usuario.

Correo electrónico Dirección de correo electrónico.

Contraseña Contraseña para el login del usuario.

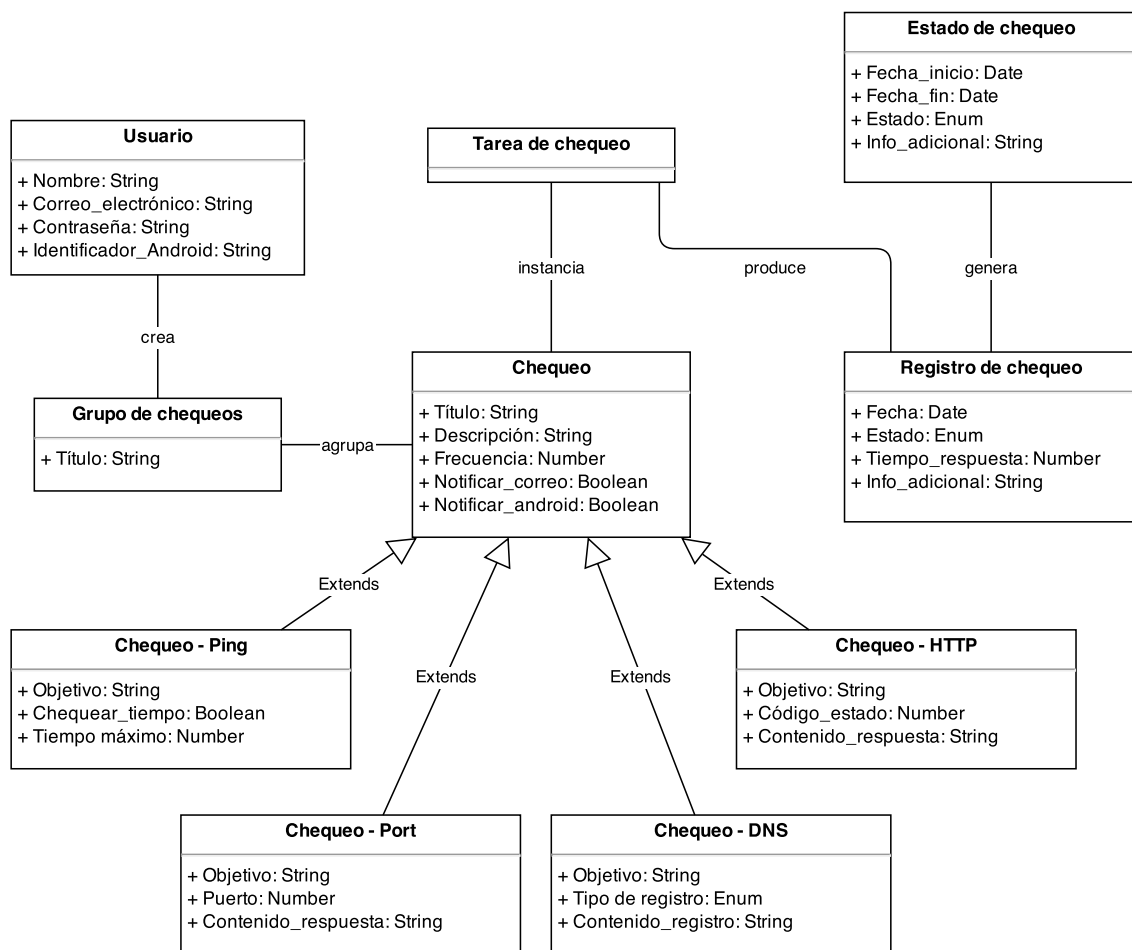


Figura 4.1: Modelo conceptual

Identificador Android Identificador único del dispositivo Android del usuario.

Relaciones

- Un **usuario** puede crear **grupos de chequeos**.

4.1.2. Grupo de chequeos

Representa a una agrupación de chequeos relacionados con los que operan de forma conjunta.

Título Título para identificar al grupo de chequeos.

Relaciones

- Un **grupo de chequeos** pertenece a un **usuario**.
- Un **grupo de chequeos** agrupa un conjunto de **chequeos**.

4.1.3. Chequeo

Representa un chequeo o punto de vigilancia sobre un servicio remoto.

Título Título para identificar al chequeo.

Descripción Información adicional en forma textual sobre el chequeo.

Frecuencia de chequeo Frecuencia temporal del chequeo, expresada en minutos.

Notificar por correo Indica si los cambios de estado del chequeo deben notificarse por correo electrónico.

Notificar por Android Indica si los cambios de estado del chequeo deben notificarse por la aplicación Android.

Relaciones

- Un **chequeo** pertenece a un **grupo**.
- Un **chequeo** lanza periódicamente **tareas de chequeo**.
- De la ejecución de un **chequeo** se generan **registros de chequeo**.
- De los cambios de estado en un **chequeo** se generan **registros de estado de chequeo**.

4.1.4. Registro de chequeo

Representa el resultado de la ejecución de un chequeo en un momento dado.

Fecha Fecha y hora en la que se obtuvo este registro.

Estado Resultado del lanzamiento del chequeo. El estado puede ser *Up*, que indica que el objetivo está funcionando; *Down*, que indica que el objetivo no está funcionando correctamente; y *Error*, que indica que hubo un problema lanzando el chequeo.

Tiempo de respuesta (solo para chequeos tipo *Ping*) indica el tiempo de respuesta medio obtenido al lanzar el chequeo.

Información adicional Datos extra en caso de que el estado no sea *Up*.

Relaciones

- Es generado por una **tarea de chequeo** y pertenece a un **chequeo**.
- Puede dar lugar al registro de un **estado de chequeo**.

4.1.5. Estado de chequeo

Representa un cambio de estado de un chequeo, detectado tras analizar un registro de chequeo.

Fecha de inicio La fecha y hora en la que el chequeo entró en este estado.

Fecha de fin La fecha y hora en la que el chequeo salió de este estado.

Estado Definición del estado del chequeo. El estado puede ser *Up*, que indica que el objetivo está funcionando; *Down*, que indica que el objetivo no está funcionando correctamente; y *Error*, que indica que hubo un problema lanzando el chequeo.

Información adicional Información adicional sobre el estado.

Relaciones

- Un **estado de chequeo** pertenece a un **chequeo**, y se genera cuando en un **registro de chequeo** y se detecta un cambio de estado.

4.1.6. Chequeo - Ping

Representa un chequeo mediante paquetes ICMP.

Objetivo Identificador de la máquina a la que enviar el ping. Debe ser un *hostname* o una IP.

Chequear tiempo? Indica si además de chequear que la máquina responde al ping, también se debe chequear que el tiempo de respuesta sea menor que el indicado.

Tiempo máximo Tiempo de respuesta máximo permitido.

Relaciones

- Extiende a un **chequeo**

4.1.7. Chequeo - Port

Representa un chequeo de un puerto en un servidor remoto.

Objetivo Identificador de la máquina a chequear.

Puerto objetivo Puerto remoto en la máquina que se chequeará.

Contenido de respuesta (*opcional*) Cadena de caracteres que deberá estar presente en la respuesta obtenida de la máquina remota.

Relaciones

- Extiende a un **chequeo**

4.1.8. Chequeo - DNS

Representa un chequeo de registros DNS sobre un dominio.

Objetivo Dominio sobre el que realizar el chequeo de DNS.

Tipo de registro Tipo de registro DNS a verificar. Puede ser A, AAAA, CNAME, MX y TXT.

Contenido de registro El contenido que debe tener el registro a consultar.

4.1.9. Chequeo - HTTP

Representa un chequeo mediante peticiones HTTP sobre un host remoto.

Objetivo URL a verificar.

Código de estado Código que debe devolver la petición a la URL indicada.

Cadena de respuesta (*opcional*) Cadena que debe encontrarse en la respuesta obtenida.

Relaciones

- Extiende a un **chequeo**

4.2. Modelo de casos de uso

El modelo de casos de uso representa las interacciones entre los actores y el sistema, desarrollado a partir de los requisitos descritos anteriormente.

4.2.1. Actores

En este apartado se describen los diversos roles que juegan los usuarios del sistema. Se reflejan todos ellos en el diagrama 4.2.

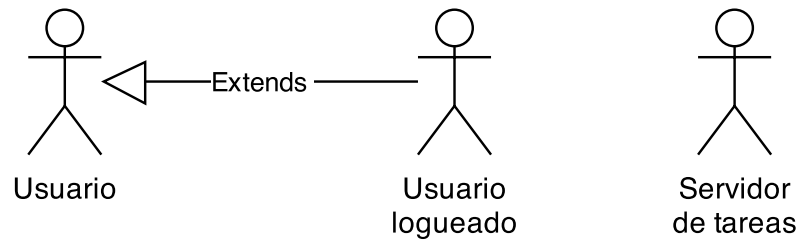


Figura 4.2: Diagrama de actores

Usuario

	Usuario
Descripción	Usuario libre de la aplicación, que puede haber hecho login previamente o no.

Usuario logueado

	Usuario logueado
Descripción	Usuario que previamente se ha registrado y ha hecho login en el sistema.

Servidor de tareas

	Servidor de tareas
Descripción	Servidor de tareas asíncronas que periódicamente lanza tareas que previamente se hayan dado de alta.

4.2.2. Subsistema de gestión de usuarios

Los casos de uso pertenecientes a este subsistema pueden verse en la figura 4.3

Caso de uso: registro de usuario

Descripción Un usuario no logueado se registra en la aplicación para poder dar de alta cheques.

Actores *Usuario.*

Escenario principal

1. Un usuario decide registrarse en el sistema y accede al panel de registro.
2. El *usuario* introduce su nombre de usuario, dirección de correo electrónico y contraseña.
3. El *sistema* comprueba que los datos introducidos son correctos.
4. El *sistema* registra al usuario.

Flujo alternativo

- 3a** Alguno de los datos introducidos es incompleto o no es correcto. El sistema informa al usuario del error.
- 3b** El nombre de usuario o dirección de correo electrónico ya han sido usados por otro usuario previamente. El sistema informa al usuario del error.

Caso de uso: login de usuario

Descripción Un usuario no logueado pero previamente registrado quiere hacer login en la aplicación.

Actores *Usuario.*

Escenario principal

1. Un usuario decide hacer login en el sistema.
2. El *usuario* accede al panel de login e introduce su nombre de usuario y contraseña.
3. El *sistema* comprueba que los datos introducidos son correctos.
4. El *sistema* loguea al usuario

Flujo alternativo

- 3a** Alguno de los datos introducidos no tiene el formato correcto o está en blanco. El sistema informa al usuario del error.
- 3b** Los datos introducidos no corresponden a ningún usuario registrado. El sistema informa al usuario del error.

Caso de uso: edición de usuario

Descripción Un usuario logueado decide editar alguno de sus datos personales.

Actores *Usuario logueado.*

Escenario principal

1. El *usuario* accede a su perfil de usuario a través del enlace *Your profile*.
2. El *sistema* muestra los datos actuales.
3. El *usuario* edita los datos que quiera y envía el formulario.
4. El *sistema* comprueba los nuevos datos y guarda los cambios.

Flujo alternativo

- 3a** Alguno de los nuevos datos no son válidos. El sistema informa del error al usuario.

4.2.3. Subsistema de gestión de grupos de chequeos

Los casos de uso pertenecientes a este subsistema pueden verse en la figura 4.4

Caso de uso: creación de grupo de chequeos

Descripción Un usuario logueado decide crear un grupo de chequeos.

Actores *Usuario logueado.*

Escenario principal

1. El *usuario* decide crear un nuevo grupo de chequeos.
2. El *usuario* pulsa en el botón de *Añadir grupo*.
3. El *sistema* muestra el formulario para añadir nuevo grupo.
4. El *usuario* escribe el título del grupo.
5. El *sistema* valida el título introducido.
6. El *sistema* crea el nuevo grupo.

Flujo alternativo

- 5a** El título introducido está en blanco o es inválido. El sistema informa al usuario del error.

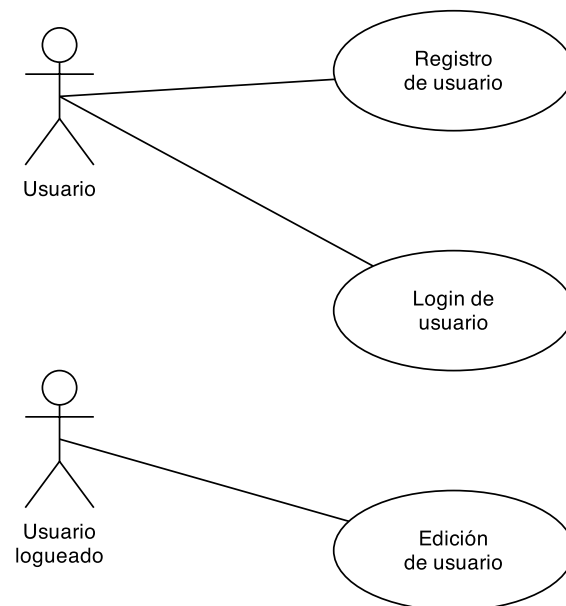


Figura 4.3: Casos de uso del subsistema de gestión de usuarios

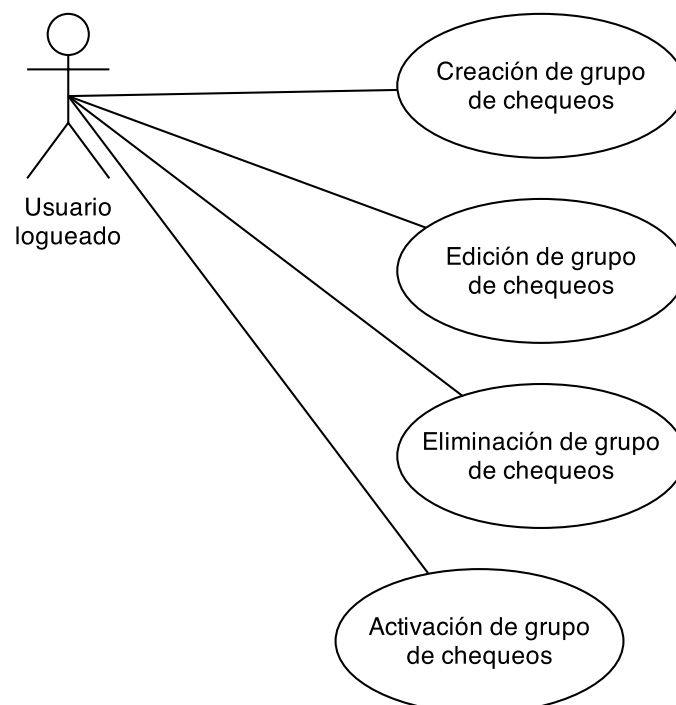


Figura 4.4: Casos de uso del subsistema de gestión de grupos de chequeos

Caso de uso: edición de grupo de chequeos

Descripción Un usuario logueado decide editar un grupo de chequeos.

Actores *Usuario logueado.*

Escenario principal

1. El *usuario* decide editar un nuevo grupo de chequeos.
2. El *usuario* pulsa en el botón de *Editar grupo* del grupo que quiera editar.
3. El *sistema* muestra el formulario para editar el grupo.
4. El *usuario* modifica el título del grupo.
5. El *sistema* valida el título introducido.
6. El *sistema* actualiza los datos del grupo.

Flujo alternativo

- 5a** El título introducido está en blanco o es inválido. El sistema informa al usuario del error.

Caso de uso: eliminación de grupo de chequeos

Descripción Un usuario logueado decide eliminar un grupo de chequeos.

Actores *Usuario logueado.*

Escenario principal

1. El *usuario* decide eliminar uno de sus grupos de chequeos.
2. El *usuario* pulsa en el botón de eliminar grupo.
3. El *sistema* muestra el formulario para confirmar la eliminación.
4. El *usuario* confirma la eliminación.
5. El *sistema* elimina los chequeos asociados al grupo.
6. El *sistema* elimina el grupo de chequeos.

Flujo alternativo

- 3a** El *usuario* no confirma la eliminación. El *sistema* muestra la pantalla inicial.

Caso de uso: activación de grupo de chequeos

Descripción Un usuario logueado decide activar un grupo de chequeos.

Actores *Usuario logueado*

Escenario principal

1. El *usuario* decide activar un grupo de chequeos.
2. El *usuario* pulsa en el botón de *Activar grupo*.
3. El *sistema* itera por todos los chequeos pertenecientes al grupo, activando cada uno de ellos.
4. El *sistema* informa al usuario de que todos los chequeos han sido activados.

Flujo alternativo

- 3a El grupo no cuenta con chequeos. El *sistema* no activa ningún chequeo.

4.2.4. Subsistema de gestión de chequeos

Los casos pertenecientes a este subsistema pueden verse en la figura 4.5.

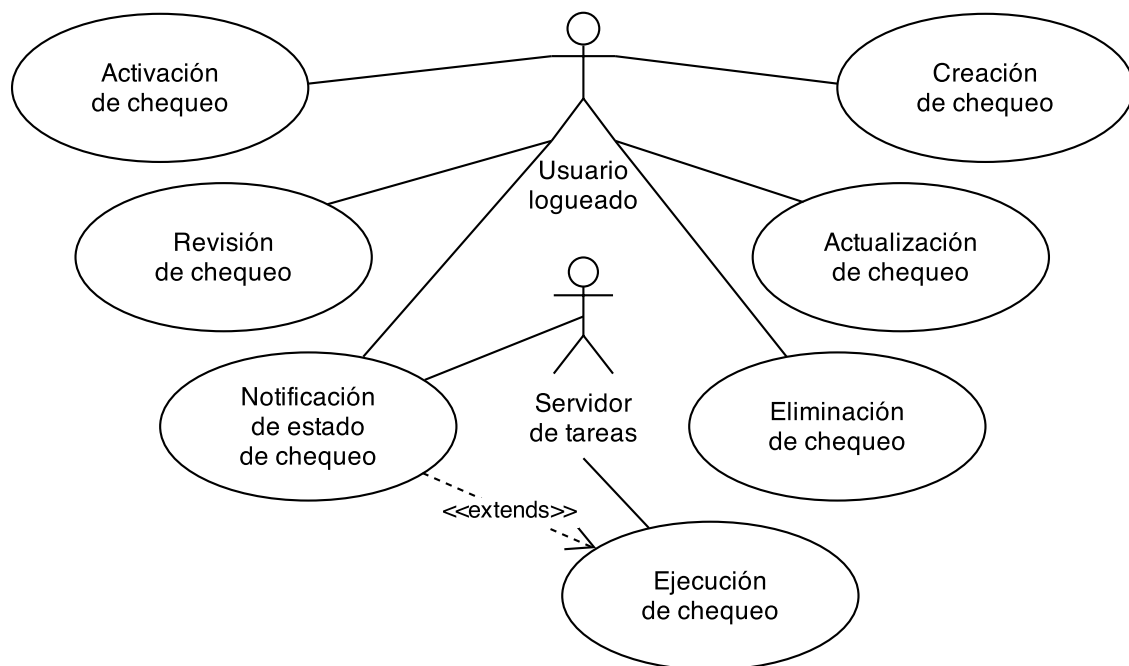


Figura 4.5: Casos de uso del subsistema de gestión de chequeos

Caso de uso: creación de chequeo

Descripción Un usuario logueado decide crear un chequeo dentro de un grupo.

Actores *Usuario logueado*

Escenario principal

1. El *usuario* decide crear un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Añadir chequeo* de un grupo.
3. El *sistema* muestra el formulario de elección del tipo de chequeo.
4. El *usuario* elige el tipo de chequeo a añadir.
5. El *sistema* muestra el formulario con los campos según el tipo de chequeo elegido.
6. El *usuario* introduce los detalles del chequeo y envía el formulario.
7. El *sistema* recibe y comprueba que los datos son correctos.
8. El *sistema* da de alta el nuevo chequeo.

Flujo alternativo

- 7a Alguno de los datos introducidos no es correcto. El *sistema* informa del error al usuario y muestra el formulario de nuevo.

Caso de uso: actualización de chequeo

Descripción Un usuario logueado decide editar un chequeo previamente creado..

Actores *Usuario logueado*

Escenario principal

1. El *usuario* decide editar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Editar chequeo*.
3. El *sistema* muestra el formulario con los campos rellenos con la información del chequeo elegido..
4. El *usuario* modifica los detalles del chequeo y envía el formulario.
5. El *sistema* recibe y comprueba que los datos son correctos.
6. El *sistema* actualiza el chequeo.

Flujo alternativo

- 5a Alguno de los datos introducidos no es correcto. El *sistema* informa del error al usuario y muestra el formulario de nuevo.

Caso de uso: eliminación de chequeo

Descripción Un usuario logueado decide eliminar un chequeo previamente creado..

Actores *Usuario logueado*

Escenario principal

1. El *usuario* decide eliminar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Eliminar chequeo*.
3. El *sistema* muestra el formulario de confirmación de eliminación
4. El *usuario* confirma la eliminación del chequeo.
5. El *sistema* elimina el chequeo.

Flujo alternativo

- 4a El *usuario* no confirma la eliminación del chequeo. El *sistema* vuelve a la pantalla inicial.

Caso de uso: activación de chequeo

Descripción Un usuario logueado decide activar un chequeo.

Actores *Usuario logueado*

Escenario principal

1. El *usuario* decide activar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *activar chequeo*.
3. El *sistema* activa el chequeo.

Caso de uso: ejecución de chequeo

Descripción El servidor de tareas tiene que lanzar un chequeo periódico.

Actores *Servidor de tareas*

Escenario principal

1. El *servidor de tareas* pide los chequeos activos.
2. El *sistema* devuelve los chequeos activos.
3. El *servidor de tareas* elige un chequeo activo.
4. El *servidor de tareas* ejecuta el chequeo.
5. El *sistema* devuelve el resultado del chequeo.
6. El *servidor de tareas* comprueba el resultado del chequeo, guardándolo en la base de datos y, si difiere del estado anterior del chequeo, lanzando el *Caso de uso: notificación de estado de chequeo*.

Flujo alternativo

- 2a** No hay chequeos activos. El *sistema* devuelve una lista vacía. El *servidor de tareas* permanece en reposo.

Caso de uso: notificación de estado de chequeo

Descripción El servidor de tareas detecta un cambio de estado y lanza notificación.

Actores *Servidor de tareas*

Escenario principal

1. El *servidor de tareas* detecta un cambio de estado de un chequeo.
2. El *servidor de tareas* comprueba el campo *Notificar_correo* del chequeo.
3. El *servidor de tareas* lanza una notificación por correo electrónico.
4. El *sistema* envía la notificación al usuario dueño del chequeo.
5. El *servidor de tareas* comprueba el campo *Notificar_android* del chequeo.
6. El *servidor de tareas* lanza una notificación Android.
7. El *sistema* envía la notificación al usuario dueño del chequeo.

Flujo alternativo

- 2a** El chequeo no tiene activa la opción *Notificar_correo*. El *servidor de tareas* no envía la notificación por correo.
- 5a** El chequeo no tiene activa la opción *Notificar_android*. El *servidor de tareas* no envía la notificación por Android.
- 5b** El usuario no tiene dispositivo Android dado de alta. El *servidor de tareas* no envía la notificación por Android.

4.3. Modelo de interfaz de usuario

En esta sección se presenta una aproximación de la navegación y apariencia de la interfaz de usuario, tanto de la plataforma web como de la aplicación móvil.

4.3.1. Modelo de navegación

El modelo de navegación de la aplicación web se presenta en la figura 4.6. El modelo de navegación de la aplicación Android se presenta en la figura 4.7

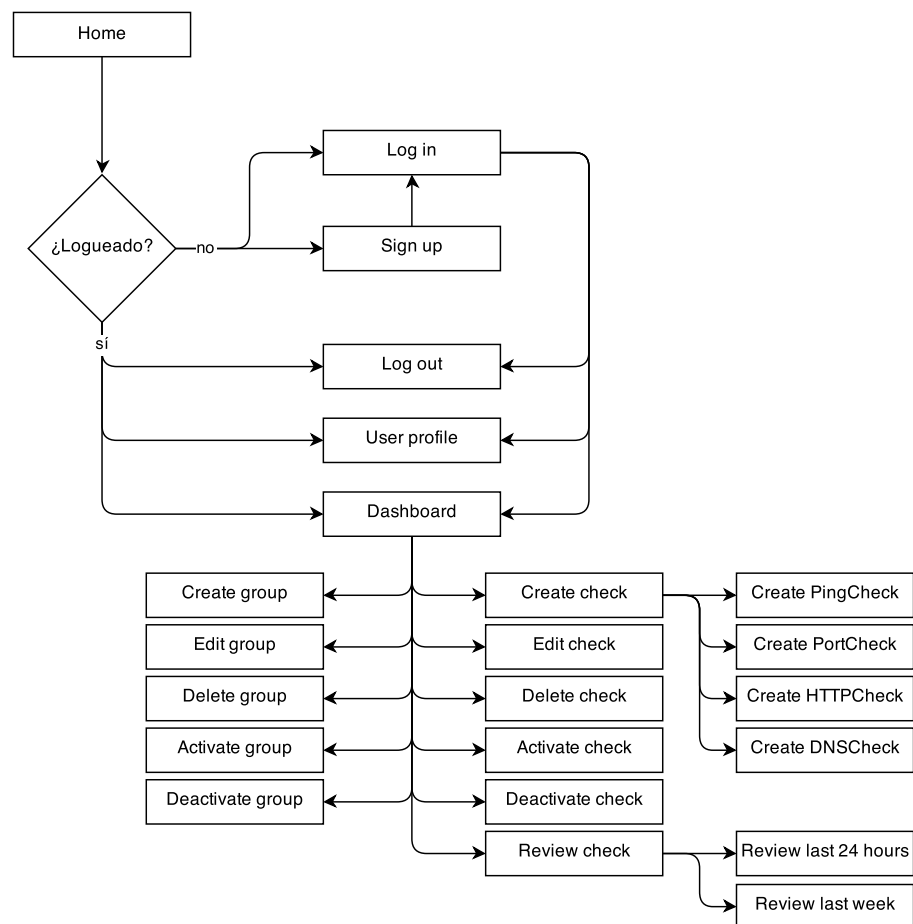


Figura 4.6: Modelo de navegación de la aplicación web

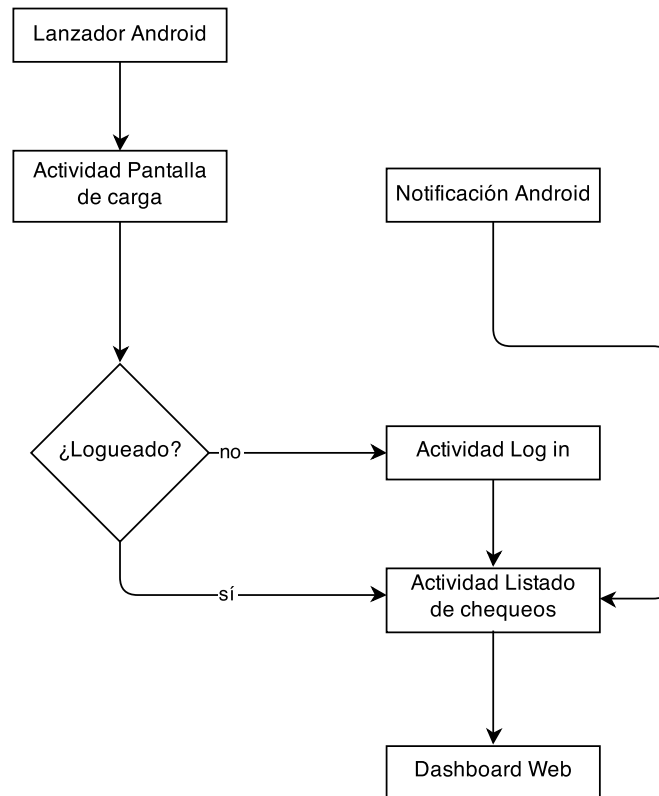


Figura 4.7: Modelo de navegación de la aplicación Android

4.3.2. Prototipos visuales de la interfaz web

A continuación se presentan los prototipos de las pantallas de la aplicación web. En concreto:

- La figura 4.8 representa la pantalla de bienvenida.
- La figura 4.9 representa la pantalla con el formulario para iniciar sesión.
- La figura 4.10 representa la pantalla para recuperar la contraseña.
- La figura 4.11 representa la pantalla para crear una cuenta de usuario.
- La figura 4.12 representa la pantalla de dashboard, en la que se ve la lista de chequeos.
- La figura 4.13 representa la pantalla de detalle de chequeo.
- La figura 4.14 representa la pantalla de creación de nuevo grupo de chequeos.
- La figura 4.15 representa la pantalla de creación de nuevo chequeo.

4.3.3. Prototipos visuales de la aplicación móvil

Seguidamente se presentan los prototipos de las interfaces visuales de la aplicación móvil. En particular:

- La figura 4.16 representa la pantalla de carga de la aplicación Android.
- La figura 4.17 representa la pantalla de login.
- La figura 4.18 representa la pantalla de lista de chequeos.
- La figura 4.19 representa una notificación recibida en el dispositivo.

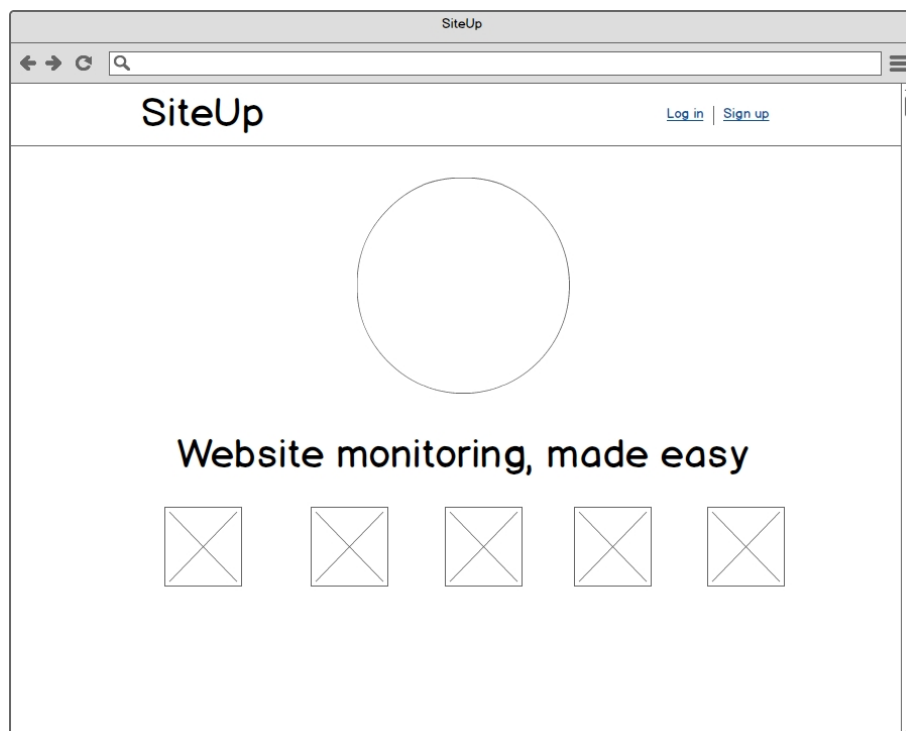
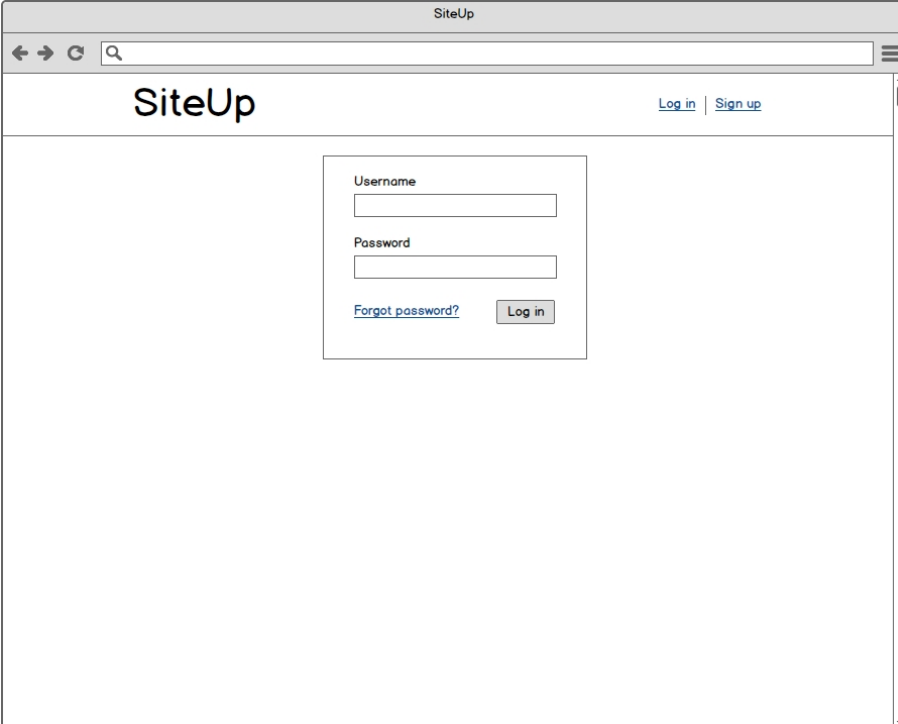


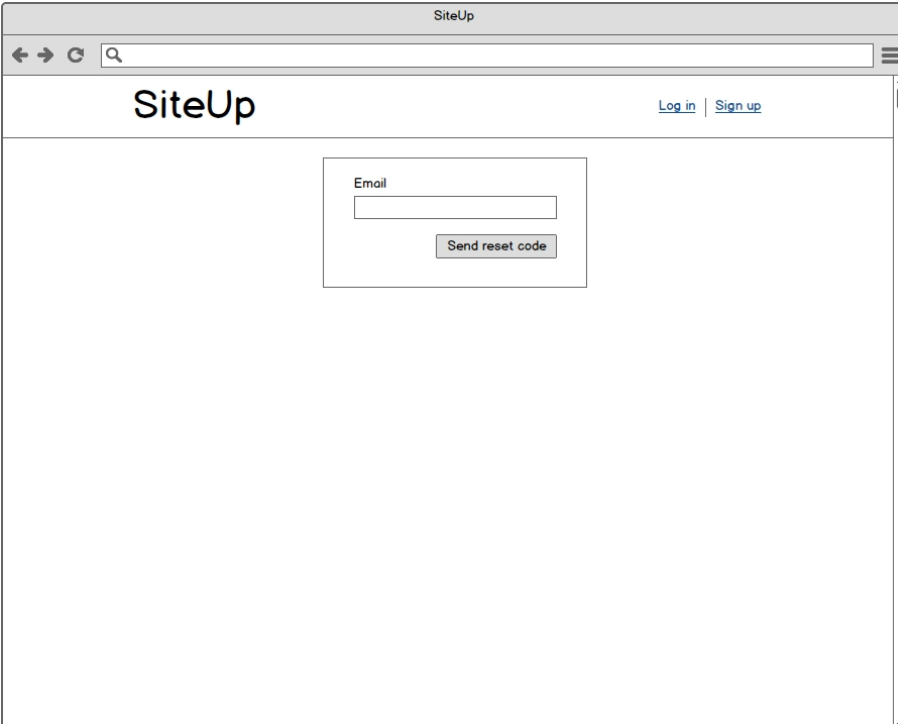
Figura 4.8: Home



The screenshot shows a web browser window with the title "SiteUp". The address bar is empty. The page header features the "SiteUp" logo on the left and "Log in" and "Sign up" links on the right. The main content area contains a login form with the following elements:

- A "Username" label above a text input field.
- A "Password" label above a text input field.
- A "Forgot password?" link below the password field.
- A "Log in" button to the right of the "Forgot password?" link.

Figura 4.9: Login



The screenshot shows the same web browser window as Figure 4.9, but the main content area displays a password recovery form with the following elements:

- An "Email" label above a text input field.
- A "Send reset code" button below the email input field.

Figura 4.10: Recuperación de contraseña

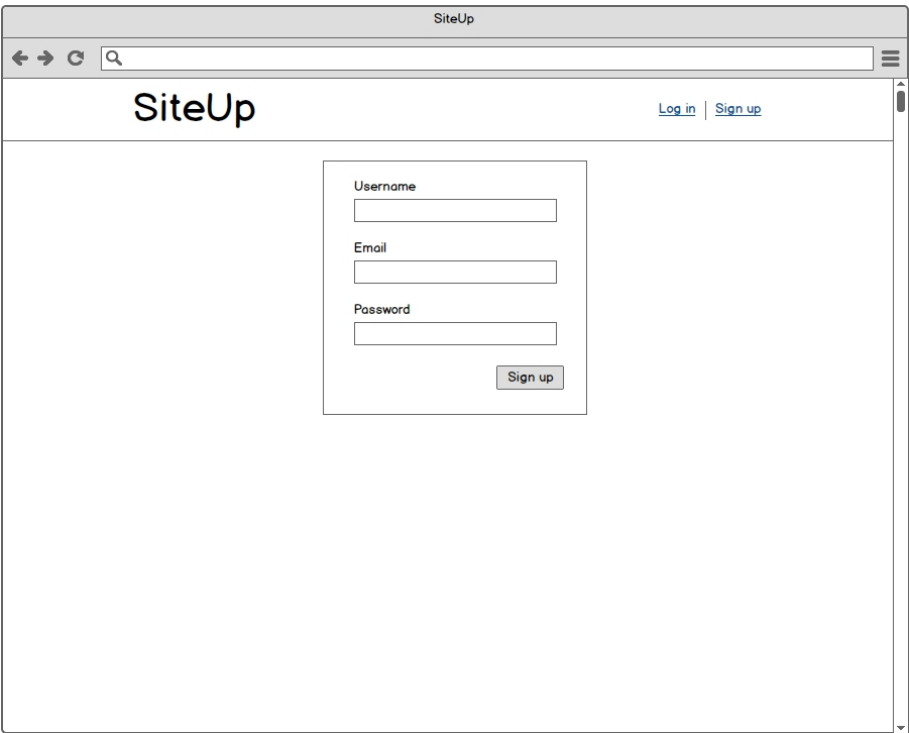


Figura 4.11: Registro de usuario

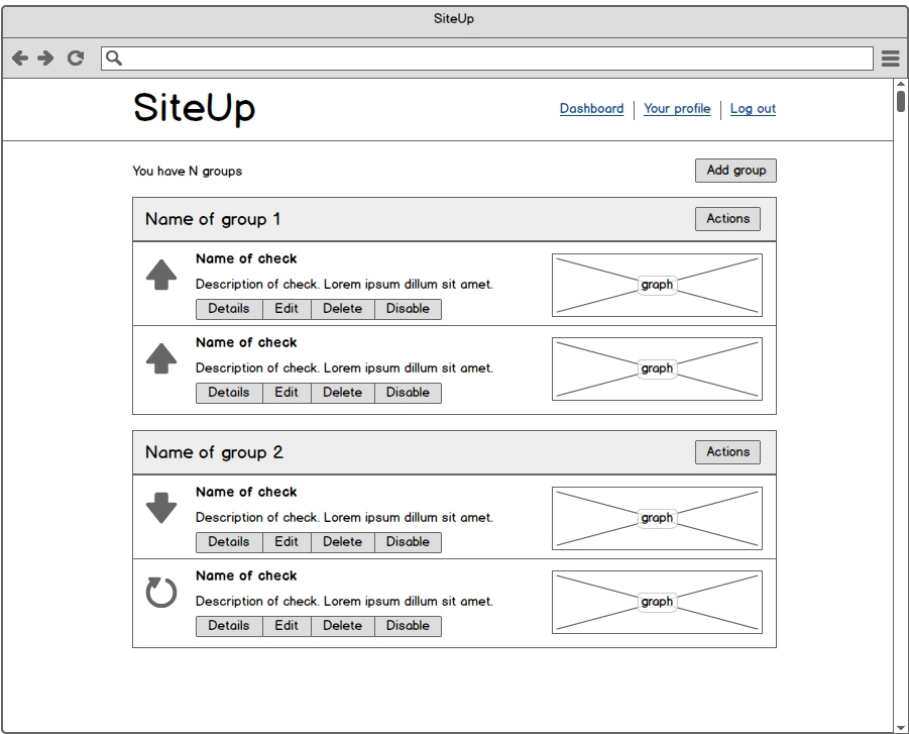


Figura 4.12: Dashboard

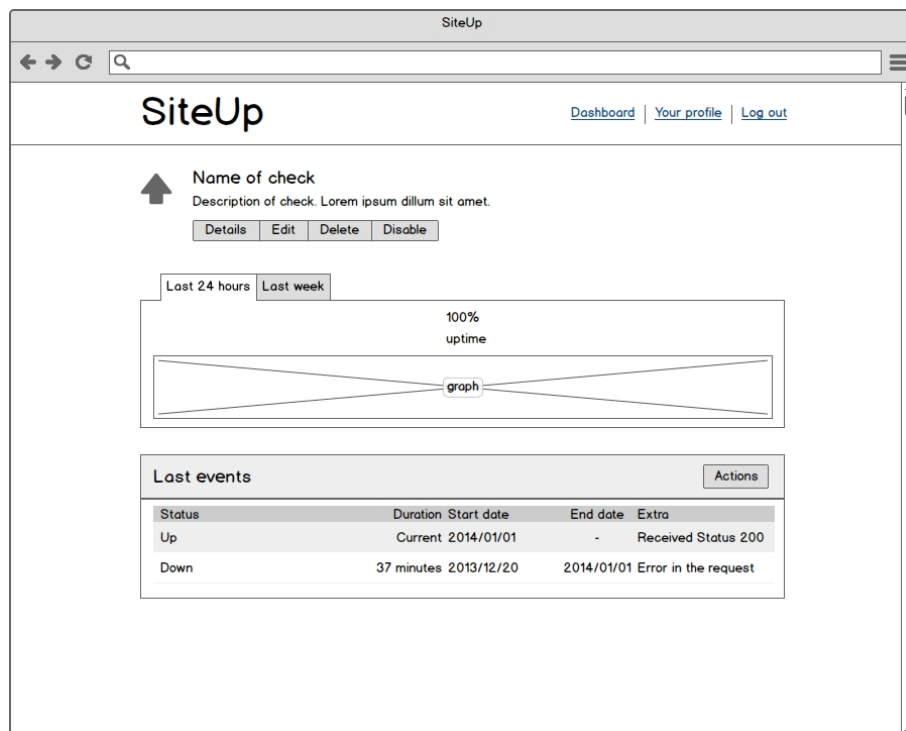


Figura 4.13: Detalle de chequeo

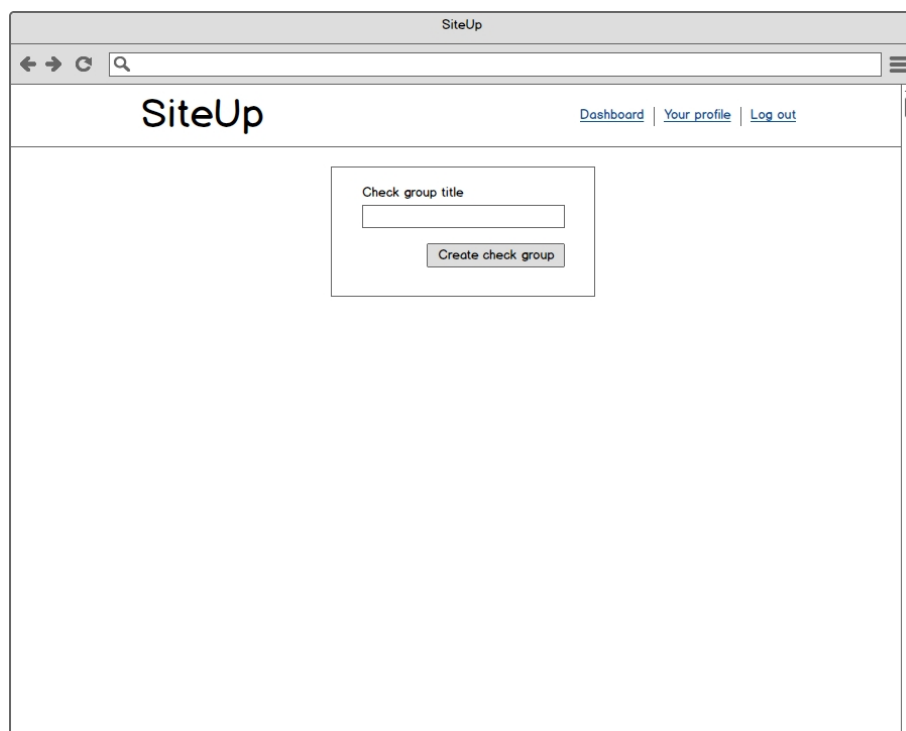
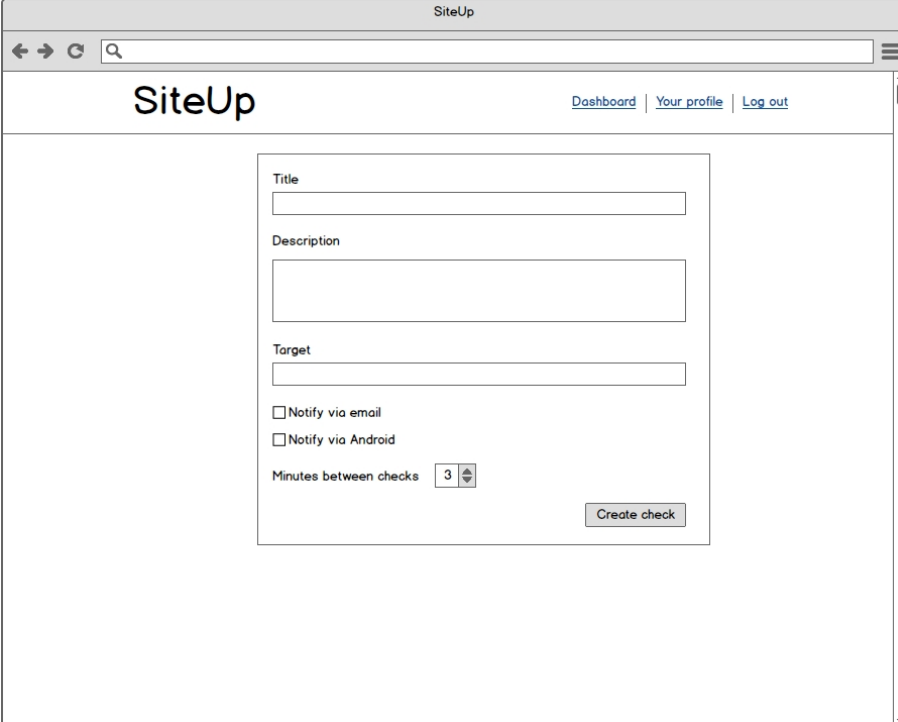


Figura 4.14: Creación de grupo de chequeos



The screenshot shows a web browser window with the title 'SiteUp'. The browser's address bar is empty. The page header includes the 'SiteUp' logo and navigation links: 'Dashboard', 'Your profile', and 'Log out'. The main content area contains a form for creating a check. The form has the following fields and controls:

- Title:** A single-line text input field.
- Description:** A multi-line text input field.
- Target:** A single-line text input field.
- Notify via email:** A checkbox.
- Notify via Android:** A checkbox.
- Minutes between checks:** A numeric input field with a spinner, currently set to 3.
- Create check:** A button.

Figura 4.15: Formulario genérico de creación de chequeo

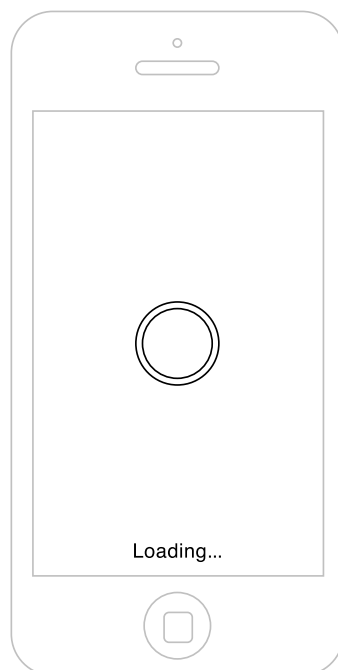


Figura 4.16: Pantalla de carga

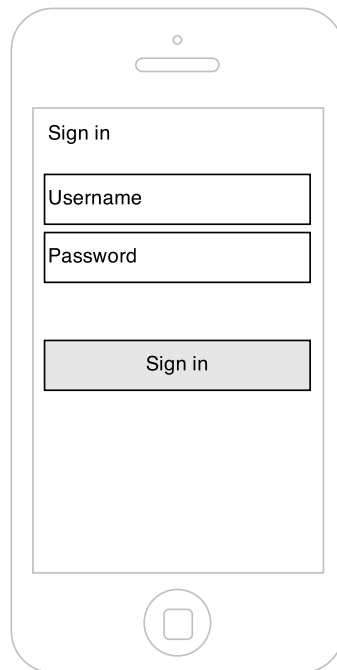


Figura 4.17: Pantalla de login

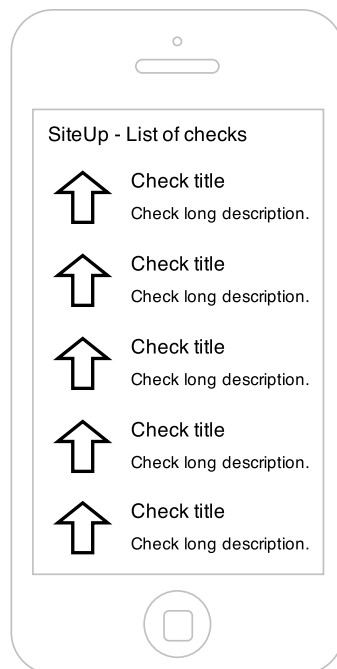


Figura 4.18: Pantalla de listado de chequeos

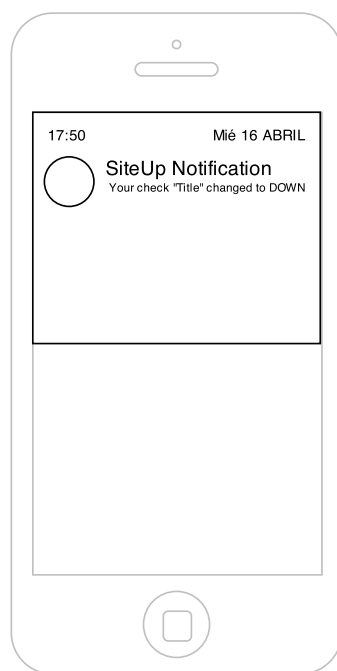


Figura 4.19: Pantalla de notificación

Capítulo 5

Diseño

En este capítulo se presentan los detalles de diseño del proyecto, basándonos en el análisis mostrado en los anteriores apartados. Se detallan la arquitectura general y el diseño físico de datos entre otros aspectos.

5.1. Arquitectura del sistema

5.1.1. Arquitectura física

En este apartado, describimos los principales componentes hardware que forman la arquitectura física de nuestro sistema, recogiendo por un lado los componentes del servidor de producción y, por otro lado, los componentes del cliente de acceso.

Servidor de producción

Hardware El hardware mínimo indispensable para la correcta ejecución del motor del proyecto se detalla en la siguiente lista:

- 512MiB de memoria RAM como mínimo.
- 10GiB de disco duro como mínimo.
- Acceso a Internet con un canal de subida de al menos 1Mbit/s.

Software En cuanto al software necesario para la ejecución del proyecto, se detallan los siguientes elementos, que es necesario instalar para desplegar el sistema:

- Sistema operativo **GNU/Linux**, preferiblemente basado en paquetería Debian.
- Código fuente del proyecto **SiteUp**.

- Servidor de shell remota **SSH**, accesible desde el exterior.
- **Nginx**, servidor web trabajando en modo de proxy inverso.
- **Supervisord**, sistema para el control de procesos que trabajen en modo demonio..
- **RabbitMQ**, cola de tareas sencilla que cumple el estandar AMQP.
- Intérprete de **Python**, versión mínima 2.7.
- Soporte de entornos virtuales **VirtualEnv** para la encapsulación de dependencias.

Cliente de acceso web

El requisito no funcional NRQ-4, presente en el cuadro 3.2.3 indica que la plataforma web deberá ser accesible desde cualquier clase de dispositivo con acceso a Internet. Así pues, la única restricción disponible para los clientes es que cuenten con un navegador moderno que provea de acceso web y sea compatible con los últimos estándares web.

Cliente de acceso Android

Los clientes que quieran acceder mediante la aplicación Android deberán contar con un dispositivo que soporte como mínimo la versión 2.3 del sistema operativo Android. Además, estos dispositivos deberán tener acceso a Internet en general, y a los Google Play Services en particular.

5.1.2. Arquitectura lógica

La arquitectura lógica del sistema está formada por los elementos software (servicios, aplicaciones, librerías, frameworks, etc.) que componen el software base, más el software desarrollado para cumplir los requisitos de la aplicación. En esta sección se muestra la organización de los distintos elementos software que componen el proyecto así como la comunicación entre ellos.

Plataforma web

En la figura 5.1 se puede ver un esquema de cómo se comunican los diferentes elementos que conforma el sistema web a alto nivel. En detalle, el flujo que sigue en cada capa es el siguiente.

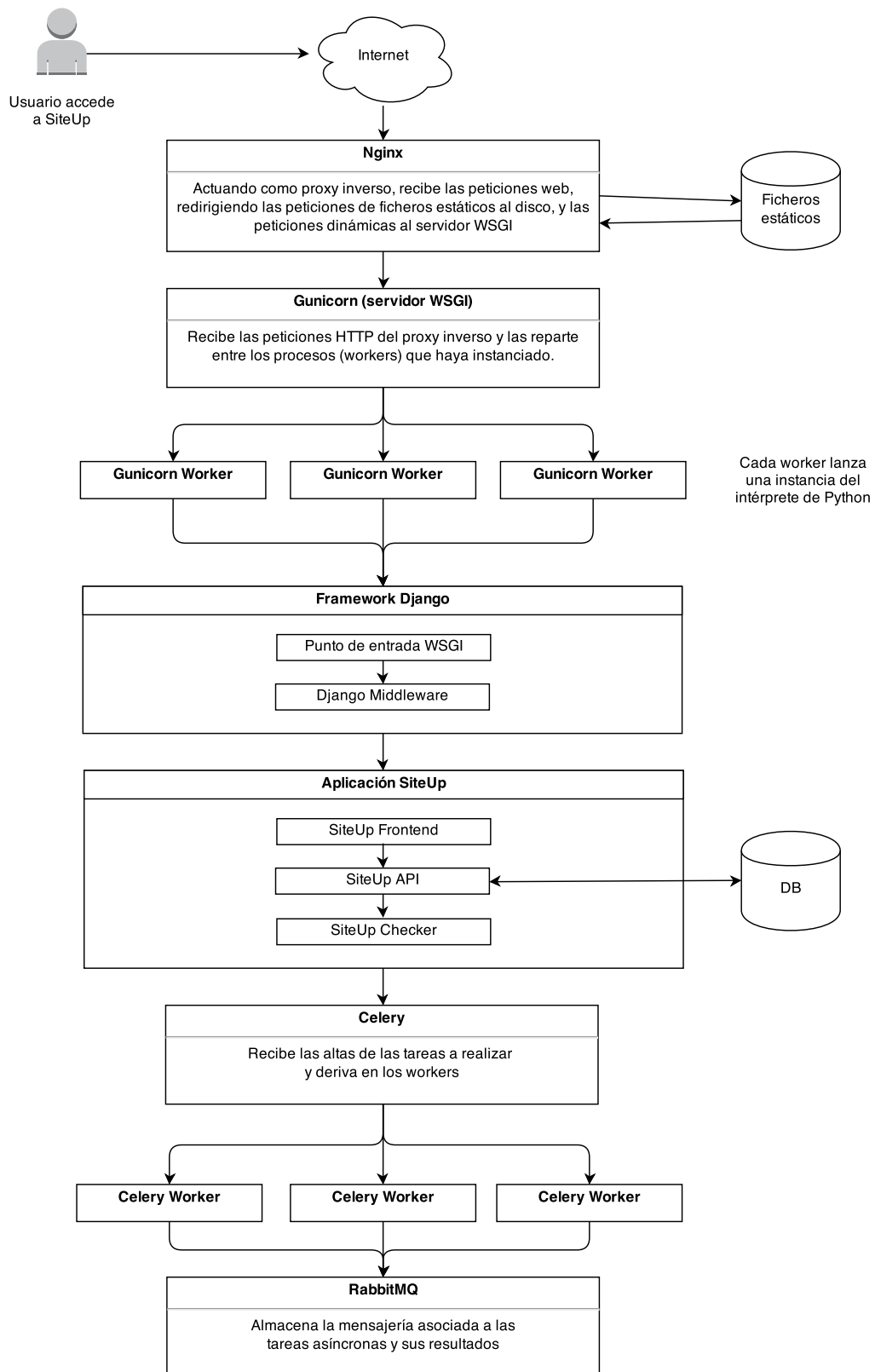


Figura 5.1: Arquitectura lógica del sistema web

Navegador Cuando un usuario desea acceder a SiteUp, abre un cliente web (habitualmente un **navegador** web) y teclea la URL. A partir de ahí, el navegador debe obtener el contenido de la web. Para ello (ignorando cuestiones de bajo nivel, como el envío de peticiones ARP o resoluciones de DNS) la acción fundamental es hacer una **petición HTTP** al servidor que se encuentra en la URL indicada. Esto se traduce en abrir una conexión TCP al puerto 80 del servidor remoto y enviar unas cabeceras en las que se indique, entre otras cosas, qué datos se quiere obtener.

Servidor proxy inverso En el servidor debe existir algún **servicio** que sea capaz de escuchar y disponer peticiones en el puerto 80 del servidor (el número el puerto puede variar). En nuestro caso el servidor deberá tener instalado **Nginx**, un servidor web/proxy inverso que recibirá las peticiones HTTP del exterior. En estas peticiones, enviadas por el navegador, se detalla qué recurso se necesita enviar.

En el caso de recursos estáticos, como por ejemplo ficheros de imágenes o archivos de hojas de estilo en cascada CSS (Cascading Style Sheet(s)), es buena práctica que sea el propio servidor el que sirva estos ficheros, dado que no necesitan de ningún procesamiento dinámico, por lo que no tiene sentido desperdiciar recursos pasando la petición al código Python.

En el caso de recursos dinámicos, esto es, las peticiones web a páginas generadas dinámicamente, Nginx actuará de proxy inverso y pasará la petición al manejador que se haya configurado.

Servidor de aplicaciones Python Como se ha comentado, el proxy inverso ha detectado que hay una petición dinámica y según su configuración la ha pasado al servidor de aplicaciones Python. En el caso de nuestro proyecto se tratará del servidor **Gunicorn**. Gunicorn sigue un modelo conocido como *pre-fork worker*, que básicamente consiste en instanciar (*forkear*) varios procesos al lanzar el servidor que se encargarán de procesar las peticiones de forma paralela, según el proceso padre las vaya repartiendo.

En el mundo del desarrollo web en Python se ha establecido un estándar, conocido como WSGI (Web Server Gateway Interface), que regula la forma en la que las aplicaciones web escritas en Python se comunican con un servidor web. Por ello, cuando un worker recibe una petición lo primero que hace es comunicarse con la interfaz WSGI de la aplicación que se vaya a ejecutar, pasándole la petición y un puntero a un *callback* al que informar cuando la respuesta a la petición esté lista.

Punto de entrada WSGI Como se ha comentado, el worker del servidor de aplicaciones se comunica con la interfaz WSGI. Todas las aplicaciones desarrolladas en Django cuentan con un módulo `wsgi.py` que actúa como punto de entrada WSGI y que directamente pasa la petición al *middleware* del framework, que empieza el procesamiento.

Framework web El framework de desarrollo elegido es **Django** [14], un proyecto de código abierto nacido en 2005. La meta fundamental de Django es facilitar la creación de sitios web complejos, haciendo especial hincapié en mantener un ritmo de desarrollo rápido y evitar la duplicidad de código. Así, gran parte del potencial de Django es la gran cantidad de funcionalidad que ya trae integrada, ya sea en el propio núcleo del framework o a través de *apps* oficiales que se incluyen en la distribución oficial.

Los elementos más destacables de Django son:

- Un sistema de mapeo objeto-relacional, que facilita enormemente el trabajo con elementos de bases de datos a través de instancias de modelos.
- Un sistema de procesamiento de peticiones a través de vistas.
- Un despachador de URLs basado en expresiones regulares.
- Soporte para plantillas.
- Un sistema de autenticación extensible.
- Una interfaz de administración que se adapta dinámicamente a cualquier proyecto.
- Un sistema de comentarios.
- Soporte integrado contra numerosos tipos de ataques vía web, como inyección SQL o cross-site scripting.

Middleware Django y aplicación A partir de este punto, la petición va rebotando entre código propio del framework Django y código de la aplicación. El flujo habitual, en su forma más básica es el siguiente:

1. Los módulos de *middleware* que gestionan las peticiones reciben la petición actual.
2. Django revisa el fichero `urls.py` de la aplicación, que contiene una lista de URLs asociadas a funciones. Django compara la URL de la petición recibida y mira si coincide con alguna de las URLs del proyecto.
3. Si no hay coincidencia, Django devuelve una respuesta negativa, normalmente en forma de código de estado 404.
4. Si hay coincidencia, Django llama a la función asociada a esa URL – conocida como la **vista**. La vista procesa la petición, devolviendo una respuesta.
5. Esa respuesta hace el camino de forma inversa, pasando por todos los pasos hasta el cliente.

Entrando más en detalle, una aplicación de Django se organiza en una arquitectura de módulos pequeños o *aplicaciones*, cada una de las cuales debe tener una serie de responsabilidades reducida y acotada, de forma que sea relativamente sencillo

intercambiar las aplicaciones por otras. En el caso del proyecto SiteUp, se cuenta con tres aplicaciones:

- **siteup_frontend** gestiona las vistas de la aplicación, recibe las peticiones, muestra las páginas y gestiona formularios.
- **siteup_api** recibe las órdenes de `siteup_frontend`, haciendo los cambios necesarios en la base de datos, dando de alta las tareas y procesando resultados.
- **siteup_checker** cuenta con el código de bajo nivel para hacer los chequeos de diferentes tipos. Es principalmente una aplicación tipo *helper*.

Cada aplicación en Django cuenta con una arquitectura de tres capas conocida como **Modelo-Plantilla-Vista**, que se asimila al clásico patrón del **Modelo-Vista-Controlador** [13].

- El **modelo** representa la información en la base de datos, evitando al usuario tener que utilizar código SQL. Para ello, Django cuenta con un ORM que facilita el trabajo con registros de la base de datos.
- Las **vistas** suelen ser funciones asociadas a una URL. La filosofía de Django es que las vistas sirvan solo como *punto de paso* de los datos entre los modelos y las plantillas. Es mala práctica colocar lógica de negocio compleja en las vistas.
- Las **plantillas** son la representación final de los datos, lo que acaba viendo el usuario. Las plantillas suelen estar escritas en HTML aunque pueden también representarse usando JSON, XML, etc.

La organización física de ficheros de un proyecto Django se ajusta perfectamente a la organización lógica hasta ahora descrita, siendo habitual tener un árbol de ficheros similar al siguiente:

- Raíz del proyecto
- `siteup` – Directorio de proyecto general.
 - `settings.py` – Configuración general.
 - `urls.py` – Configuración de URLs.
 - `wsgi.py` – Punto de acceso WSGI.
- `siteup_api` – Directorio para la app (suele haber más de uno).
 - `views.py` – Definición de vistas (funciones).
 - `models.py` – Definición de modelos.
 - `tests.py` – Definición de tests.

Opcionalmente, las aplicaciones de Django pueden contar con ficheros individuales para otros elementos, como formularios, *mánagers*, migraciones, validadores, utilidades, filtros, procesadores de contexto, etcétera.

Servidor de tareas asíncronas y bróker de mensajes Habrá peticiones en SiteUp cuyo objetivo sea dar de alta chequeos. Éstos dan lugar a nuevas tareas que serán ejecutadas asíncronamente.

Para la gestión de tareas de forma asíncrona se ha utilizado **Celery**, una cola de tareas asíncrona enfocada a operaciones en tiempo real. Celery está escrito en Python, pero se integra fácilmente con cualquier otro sistema que implemente su interfaz. Las unidades de ejecución básicas, las *tareas*, se ejecutan de forma concurrente en uno o varios *workers*. Las tareas pueden ejecutarse de forma asíncrona, en segundo plano, o de forma síncrona, haciendo que el flujo de ejecución espere hasta que el resultado de la tarea esté disponible.

Celery se comunica mediante mensajes, normalmente utilizando un **broker** que media entre los clientes y los workers. Para iniciar una tarea, el cliente pone un mensaje en la cola del broker. Entonces, el broker entrega el mensaje al worker de Celery. La opción recomendada, y la que se ha utilizado en SiteUp, es usar **RabbitMQ** como broker de mensajes.

RabbitMQ, y la mayoría de brókers de mensajes, implementan el protocolo AMQP, un estándar abierto para enviar mensajes entre aplicaciones y organizaciones, siempre que éstas sean capaces de interactuar usando las interfaces del protocolo.

Tras su conclusión, Celery se comunica con Django para dejar constancia de los resultados de esas tareas en forma de modelos de la base de datos.

Aplicación Android

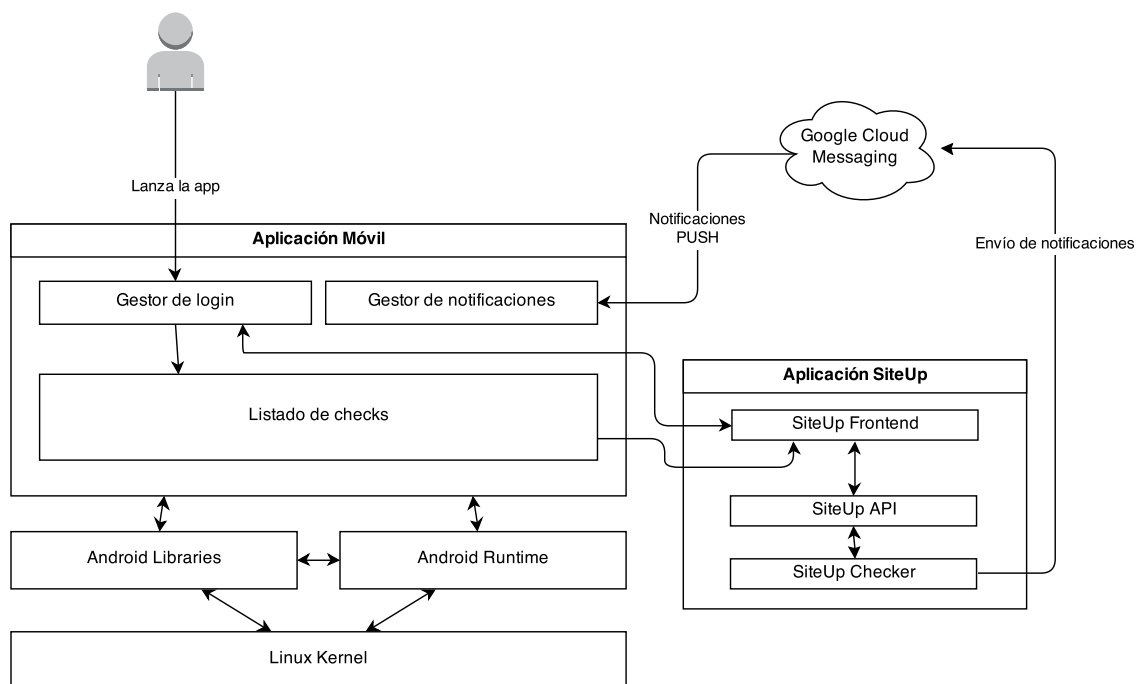


Figura 5.2: Arquitectura lógica de la aplicación Android

En la figura 5.2 se puede ver un esquema general de la arquitectura de la aplicación móvil desarrollada para el sistema operativo Android. El flujo se detalla a continuación:

Lanzador Partimos de la premisa de que el usuario ha instalado la aplicación *SiteUp Client* en su dispositivo. Si el usuario decide lanzar la aplicación, deberá dirigirse al lanzador de aplicaciones de su dispositivo y pulsar sobre el icono de la aplicación, con lo que se iniciará la ejecución.

Gestor de inicio de sesión Cuando la aplicación Android se inicia, se lanza una actividad *LoginActivity* que hace varias cosas. En primer lugar, verifica si hay datos de inicio de sesión guardados en el dispositivo. Si no los hay, muestra un formulario para que el usuario introduzca su nombre de usuario y contraseña.

Una vez obtenidos los datos, la aplicación móvil se comunica con la plataforma web para verificar el inicio de sesión. En caso correcto, la plataforma web envía la información de los chequeos a la aplicación móvil.

Listado de chequeos En la aplicación móvil se ha dispuesto una actividad *CheckListActivity* que mostrará la lista de chequeos que el usuario ha dado de alta, con información básica de cada uno de ellos. Cuando el usuario pulse en cualquiera de los chequeos, se abrirá un navegador y será redirigido a la plataforma web, particularmente a la página de detalle del chequeo seleccionado.

Gestor de notificaciones La aplicación móvil cuenta con un servicio capaz de recibir notificaciones *Push* del servicio *Google Cloud Messaging*. Cuando se recibe un mensaje, se muestra una notificación al usuario, informándole también de forma audible y a través de vibración. La notificación aparece en el área de notificaciones del sistema operativo. Si el usuario pulsa en la notificación podrá obtener información sobre el chequeo involucrado.

Google Cloud Messaging GCM [22] es un servicio para Android que permite enviar datos desde un servidor a los dispositivos Android que tengan instaladas una aplicación en particular. El servicio GCM se encarga de todos los aspectos de la entrega de mensajes. Se trata de un servicio gratuito independientemente del número de mensajes y el tamaño de éstos.

GCM recibirá las notificaciones del *backend* de la plataforma móvil, y se encargará de reenviarlas a los dispositivos Android.

SiteUp Checker Dentro de la aplicación móvil, el módulo *siteup_checker* es el encargado de realizar los chequeos, y también de enviar las notificaciones a los dispositivos Android a través del servicio GCM. La plataforma web tiene una clave

especial con la que se realizan las peticiones a los servidores de Google, enviando los mensajes.

Runtime y bibliotecas de Android Las aplicaciones Android utilizan como base el runtime y las bibliotecas que provee el sistema operativo para acceder a las funciones del dispositivo y demás servicios, como la comunicación por red, acceso a datos del usuario, lectura y escritura de ficheros en el dispositivo, etcétera.

5.2. Diseño detallado de componentes

En esta sección se detalla el diseño concreto de algunos componentes de la aplicación, especificando las clases involucradas y el flujo de mensajes entre ellas.

En particular, se han elegido el proceso de **lanzamiento y notificación de un chequeo** por ser el que mayor importancia y complejidad tiene.

5.2.1. Lanzamiento y notificación de un chequeo

El proceso de lanzamiento y notificación de un chequeo es un proceso que involucra bastantes elementos y supone un componente esencial en la plataforma web. En la figura 5.3 se puede observar el flujo de mensajes entre los diferentes elementos.

En particular, el proceso que se sigue es el siguiente. En primer lugar, desde el módulo `siteup/celery` se crea una app de **Celery**, que revisa los módulos de la aplicación y *descubre* las tareas. En particular, encuentra tareas en los módulos `siteup_checker/tasks` y `siteup_checker/tasks_notification`.

```
app = Celery('siteup')
```

```
app.config_from_object('django.conf:settings')
app.autodiscover_tasks(lambda: settings.INSTALLED_APPS)
```

Este proceso se produce una vez al lanzar el servidor en producción, por lo que no se ha reflejado en el diagrama de secuencia.

Una vez dadas de alta las tareas, Celery cuenta con un **scheduler** que ejecuta las tareas periódicas que haya encontrado en el proceso de descubrimiento anterior. Como se ha mencionado, las tareas de SiteUp se definen en los módulos `siteup_checker/tasks` y `siteup_checker/tasks_notification`. Las tareas periódicas se definen en forma de función *decorada* con una cabecera que indica su periodicidad, por ejemplo:

```
@periodic_task(run_every=crontab(hour="*", minute="*", day_of_week="*"))
def enqueue_checks():
    # ...
```

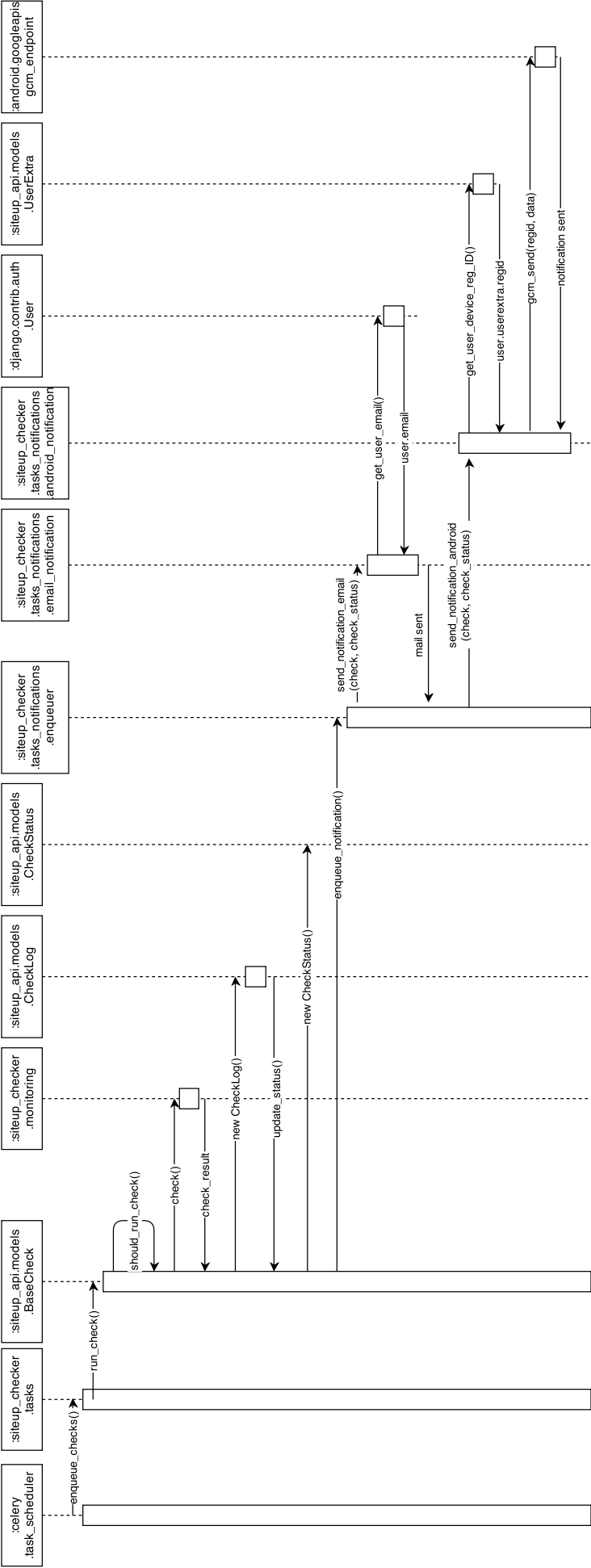


Figura 5.3: Diagrama de secuencia del proceso de lanzamiento de un chequeo

Precisamente la tarea `enqueue_checks` es la principal. Se encarga de revisar los chequeos dados de alta en el sistema y llamar al método `run_check` de cada uno de ellos. Este método se define en la clase base `BaseCheck` de la que heredan todos los modelos, y delega cierta funcionalidad a cada check particular según su tipo. Una de las funciones comunes a todos los tipos de chequeo es llamar a la función `should_run_check`, que revisa si el chequeo debe ejecutarse o no atendiendo a varias razones:

- Si ha sido recién creado (se fuerza el chequeo).
- Tiempo desde el último chequeo (según la variable `last_log_datetime`).
- Chequeo activado o desactivado (según la variable `is_active` del chequeo).

Si la verificación es positiva, el chequeo hace uso del módulo `siteup_checker.monitoring`, que contiene los procedimientos que llevan a cabo las operaciones de chequeo, según el tipo. Por ejemplo, el chequeo mediante ping lo lleva a cabo la función `check_ping` dentro del módulo `siteup_checker.monitoring.ping`.

El procedimiento de chequeo lleva a cabo el proceso según sea necesario, y devuelve un resultado que siempre cuenta con los siguientes campos:

valid Booleano que indica si el chequeo se ha podido llevar a cabo o por el contrario ha habido un error en el lanzamiento (un error del servidor de despliegue, por ejemplo).

status Entero que indica el estado del chequeo:

- 0 Chequeo válido, objetivo correcto.
- 1 Chequeo válido, objetivo caído.
- 2 Chequeo inválido.

status_extra Información adicional sobre el chequeo.

El resultado es devuelto al método `run_check`, y éste se encarga de generar un registro de chequeo, creando una instancia del modelo `CheckLog` con la información obtenida.

Al crear el `CheckLog` se llama al método `update_status` del chequeo. Éste se encarga de revisar los últimos registros generados. Si detecta que hay un cambio de estado (esto es, que el objetivo vigilado tenía previamente un estado y tras los últimos chequeos ha cambiado), se inicia el proceso de estado y notificación. En primer lugar, se crea una instancia de `CheckStatus`, que guarda información sobre el nuevo estado.

En segundo lugar, se lanza el proceso de notificación. Para ello, se llama al procedimiento `enqueue_notifications` dentro del módulo `siteup_checker.tasks_notification`, que recibe el chequeo y el nuevo estado. Desde el procedimiento se llama a la función `send_notification_email`, que lee de la base de datos el email de usuario y envía la notificación por correo electrónico.

También se llama a la función `send_notification_android`. Ésta, en primer lugar, lee de la base de datos la información del usuario, almacenada en el modelo `UserExtra`, y extra el número de registro del dispositivo, si hay alguno. Seguidamente, se genera la carga de datos (*payload*) que se va a enviar a la aplicación mediante la función `prepare_notification_android`, y finalmente se envía a los servidores del servicio de Google Cloud Messaging, utilizando la clave de API establecida en la configuración (véase la sección *Instalación de credenciales* del manual de instalación).

Tras ello, el procedimiento de chequeo se habrá completado.

5.3. Diseño físico de datos

En la figura 5.4 se muestra el diagrama de modelos que se reflejará en la base de datos, donde se puede apreciar cada una de las tablas que forman el proyecto, así como los campos de las tablas y las relaciones entre ellas.

El diseño de los datos se ha hecho mediante las herramientas de mapeo objeto-relacional que ofrece el framework Django. A continuación se detallan los principales modelos con los que cuenta el sistema.

5.3.1. User

Este modelo representa a un usuario registrado en el sistema. Es un modelo pre-determinado de Django y forma parte de la aplicación `django.contrib.auth`.

Sus principales campos son:

- `username` - Nombre de usuario.
- `password` - Contraseña. Se almacena en forma de cifrado irreversible de acuerdo a las exigencias de la actual Ley de Protección de Datos.
- `email` - Dirección de correo electrónico.

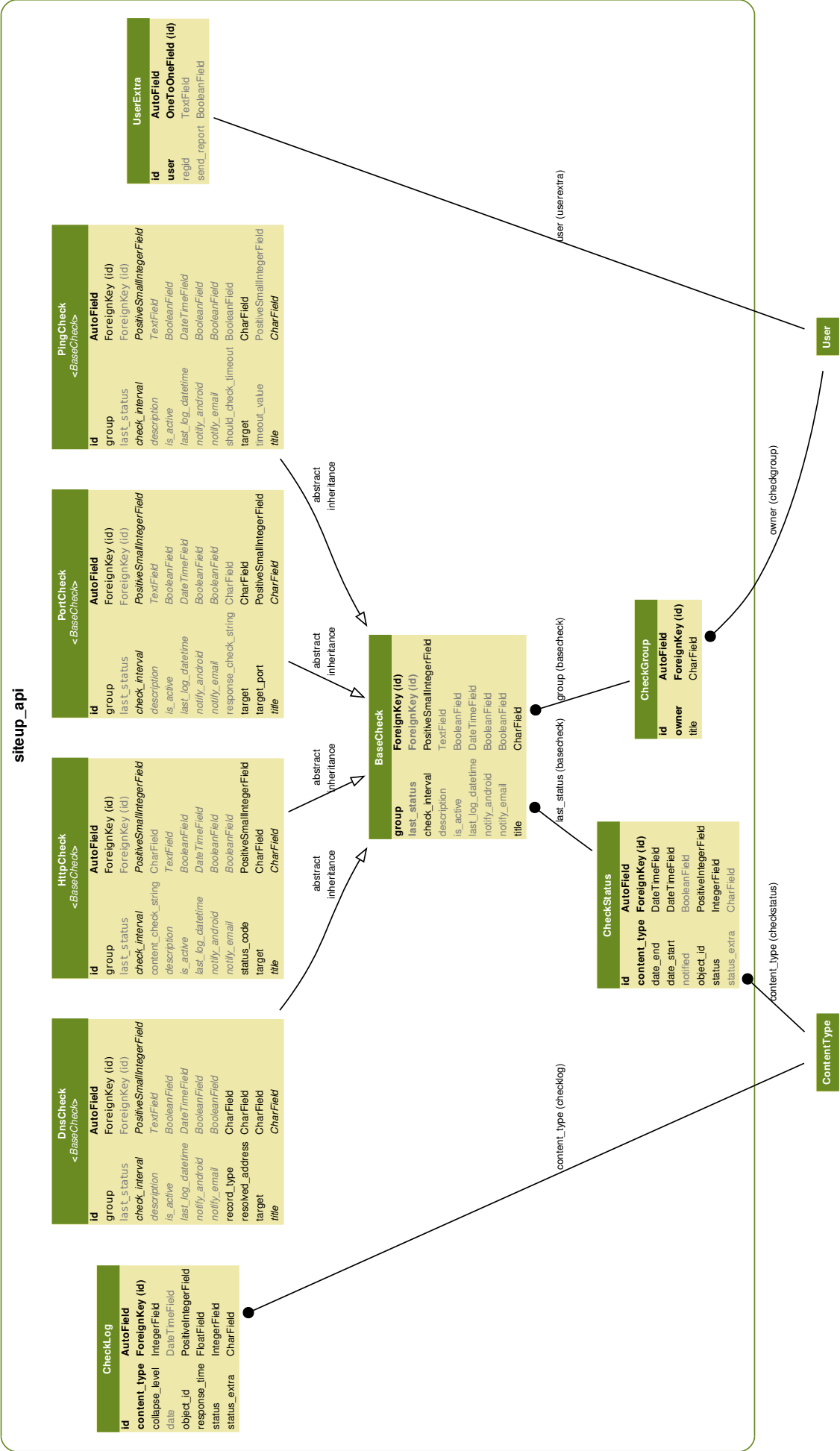
Relaciones:

Usuario - CheckGroup 1:N, relación fuerte, el usuario es la entidad fuerte.

Usuario - UserExtra 1:1, relación fuerte, el usuario es la entidad fuerte.

5.3.2. UserExtra

Este modelo guarda información adicional sobre los usuarios, evitando así tener que modificar el modelo que integra Django. Forma parte de la aplicación `siteup_api`.



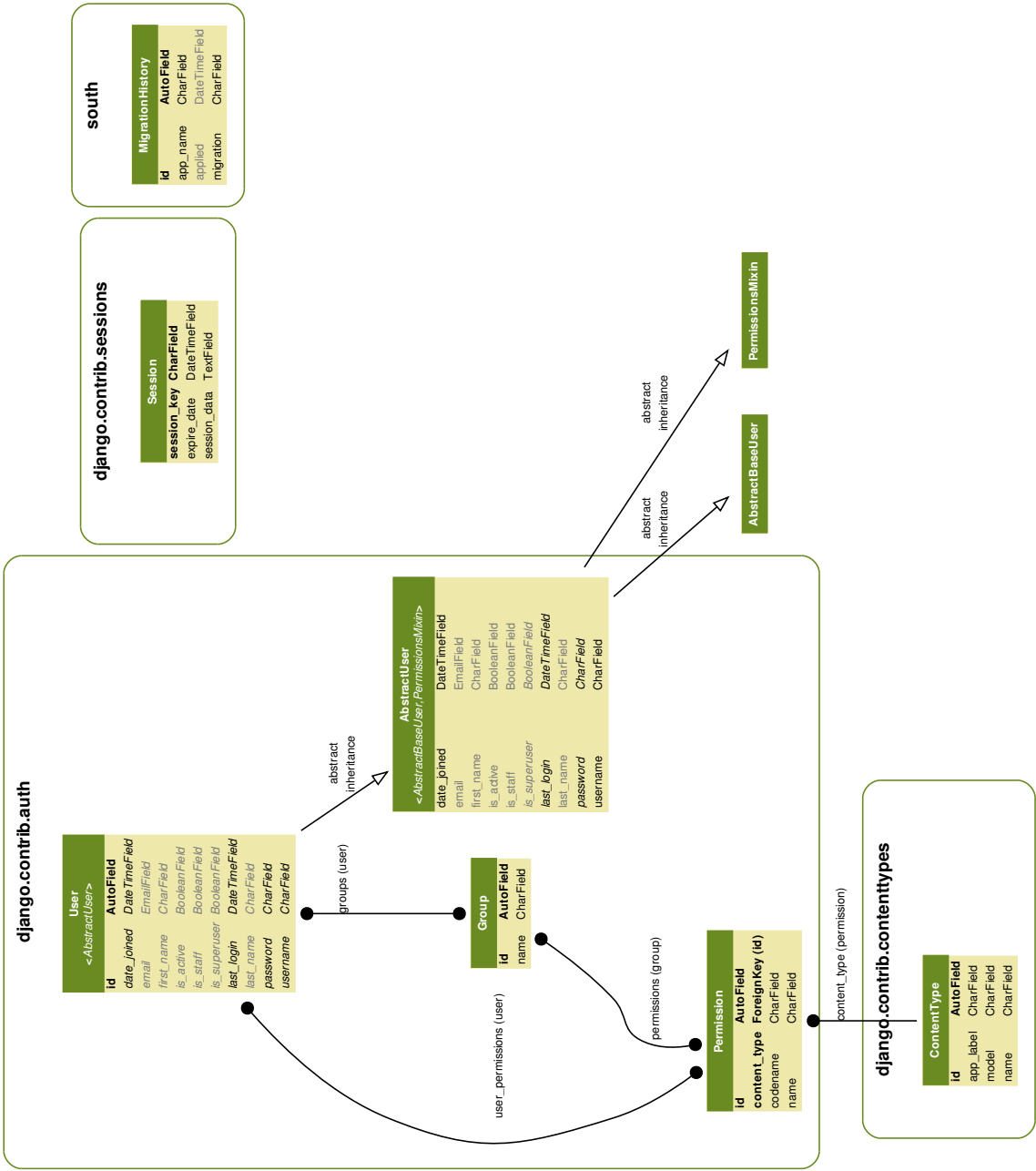


Figura 5.5: Diseño de la base de datos (II)

Sus principales campos son:

- `send_report` - Booleano, indica si el usuario recibirá reportes diarios sobre el estado de sus chequeos.
- `regid` - Texto, guarda el identificador único del dispositivo Android obtenido a través de Google Play Services.

Relaciones:

Usuario - UserExtra 1:1, relación fuerte, el usuario es la entidad fuerte.

5.3.3. CheckGroup

Parte de la aplicación `siteup_api`. Identifica a un grupo de chequeos. Todos los chequeos deben pertenecer a un grupo.

El campo principal de este modelo es `title`, que guarda una cadena con el título del grupo.

Relaciones:

Usuario - CheckGroup 1:N, relación fuerte, el usuario es la entidad fuerte.

CheckGroup - BaseCheck 1:N, relación fuerte, el grupo es la entidad fuerte.

5.3.4. BaseCheck

Parte de la aplicación `siteup_api`. Este modelo **abstracto** guarda campos comunes a todos los tipos de chequeos. No tiene representación directa en la base de datos, sino que sirve para evitar duplicidad de código. Así, todas los modelos derivados de `BaseCheck` heredan sus atributos y relaciones.

Sus principales campos son:

- `title` - Texto, representa el título del chequeo.
- `description` - Texto, representa la descripción del chequeo.
- `is_active` - Booleano, indica si el chequeo está activo.
- `check_interval` - Entero, indica el intervalo entre chequeos en minutos.
- `notify_email` - Booleano, indica si los cambios de estado del chequeo se notificarán vía correo electrónico.
- `notify_android` - Booleano, indica si los cambios de estado del chequeo se notificarán vía Android utilizando notificaciones push.
- `last_log_datetime` - Fecha, almacena la fecha de la última vez que se lanzó el chequeo.

Las relaciones de este modelo (representadas en la base de datos a través de las tablas de las clases heredadas son las siguientes:

Relaciones:

CheckGroup - BaseCheck 1:N, relación fuerte, el grupo es la entidad fuerte.

BaseCheck - CheckLog 1:N, relación fuerte, el *BaseCheck* es la entidad fuerte.

BaseCheck - CheckStatus 1:N, relación fuerte, el *BaseCheck* es la entidad fuerte.

BaseCheck - CheckStatus 1:1, relación fuerte, el *BaseCheck* es la entidad fuerte.

5.3.5. DnsCheck

Parte de la aplicación *siteup_api*. Este modelo deriva de *BaseCheck* y representa un chequeo de registros DNS. Sus principales campos son:

- **target** - Texto, indica el dominio a chequear.
- **record_type** - Enum, indica el tipo de registro a chequear. Puede ser de tipo A, AAAA, CNAME, MX y TXT.
- **resolved_address** - Texto, indica el contenido que debe tener el registro revisado.

5.3.6. HttpCheck

Parte de la aplicación *siteup_api*. Este modelo deriva de *BaseCheck* y representa un chequeo a través de peticiones HTTP.

Sus principales campos son:

- **target** - Texto, indica la URL a comprobar.
- **status_code** - Entero, indica el código de estado que se debe recibir.
- **content_check_string** - Cadena, representa una cadena de texto que, opcionalmente, debe estar presente en la respuesta obtenida desde la URL indicada.

5.3.7. PortCheck

Parte de la aplicación *siteup_api*. Este modelo deriva de *BaseCheck* y representa un chequeo de puertos remotos.

Sus principales campos son:

- **target** - Cadena, identifica el servidor remoto al que conectarse.

- `target_port` - Entero, indica el puerto remoto al que conectarse.
- `response_check_string` - Cadena, representa una cadena de texto que, opcionalmente, debe estar presente en la respuesta obtenida desde el servidor.

5.3.8. PingCheck

Parte de la aplicación `siteup_api`. Este modelo deriva de `BaseCheck` y representa un chequeo mediante el envío de paquetes ICMP.

Sus principales campos son:

- `target` - Cadena, identifica al servidor remoto que hay que chequear.
- `should_check_timeout` - Booleano, indica si hay que comprobar el tiempo de respuesta de los paquetes ping.
- `timeout_value` - Entero, si el anterior campo evalúa a *Verdadero*, este campo guarda el tiempo de respuesta máximo permitido.

5.3.9. CheckLog

Parte de la aplicación `siteup_api`. Es un registro (*log*) que representa el resultado de ejecutar un chequeo.

Sus principales campos son:

- `date` - Fecha, representa el momento en el que se obtuvo el resultado.
- `status` - Entero, representa el estado del chequeo que se ha obtenido. Los posibles valores son:
 - 0: el chequeo ha terminado correctamente y el estado es correcto (*Up*).
 - 1: el chequeo ha terminado correctamente y el estado es negativo (*Down*).
 - 2: el chequeo no ha podido concluirse. Equivalente en estadísticas al estado negativo. (*Error*).
- `response_time` - Entero, solo válido para los chequeos de tipo Ping. Guarda el tiempo de respuesta obtenido.
- `collapse_level` - Entero. Indica el nivel de *colapsado* del registro. Dado que el número de registros es muy alto (en una hora pueden llegar a generarse más de 3600 registros por chequeo), es necesario resumir los registros más antiguos. Este campo indica si el *CheckLog* ya ha sido resumido o no.

Sus relaciones son:

BaseCheck - CheckLog 1:N, relación fuerte, el *BaseCheck* es la entidad fuerte.

5.3.10. CheckStatus

Parte de la aplicación `siteup_api`. Representa un cambio de estado de un chequeo. Cuando un chequeo es ejecutado, se genera una instancia de *CheckLog* y se comprueba el estado de éste con el estado de la última comprobación. Si se verifica que el estado ha cambiado con respecto a los últimos estados, se genera una instancia de *CheckStatus* y se lanzan las notificaciones pertinentes.

Los campos principales de este modelo son:

- `status` - Entero, indica el estado. El código numérico responde al mismo formato que se sigue en el modelo *CheckLog*.
- `status_extra` - Cadena, guarda información adicional sobre el estado.
- `date_start` - Fecha, indica el momento en el que el chequeo cambia a este estado.
- `date_end` - Fecha, indica el momento en el que el chequeo deja de estar en este estado. Si es el estado actual de un chequeo, se mantiene en blanco.
- `notified` - Booleano, indica si este cambio de estado ha sido ya notificado.

Sus relaciones son:

BaseCheck - CheckLog 1:N, relación fuerte, el *BaseCheck* es la entidad fuerte.

BaseCheck - CheckStatus 1:N, relación fuerte, el *BaseCheck* es la entidad fuerte.

BaseCheck - CheckStatus 1:1, relación fuerte, el *BaseCheck* es la entidad fuerte.

5.3.11. Session

Parte de la aplicación `django.contrib.sessions`. Representa una sesión de usuario en la aplicación web. Sus campos principales son:

- `session_key` - Cadena, guarda el identificador de la sesión.
- `session_data` - Texto, guarda los datos asociados a la sesión.
- `expire_date` - Fecha, indica hasta qué fecha son válidos los datos de la sesión.

5.3.12. MigrationHistory

Parte de la aplicación `south`. Representa una migración de la base de datos. Los campos principales son:

- `app_name` - Cadena, nombre de la app sobre la que se hace la migración.
- `migration` - Cadena, identificador de la migración.

- `applied` - Fecha, indica cuándo se aplicó la migración.

5.3.13. ContentType

Parte de la aplicación `django.contrib.contenttypes`. Es el bloque principal del *Content Types Framework*, un framework integrado en Django que permite trabajar de forma genérica con los modelos, ofreciendo funcionalidades como la posibilidad de crear claves foráneas genéricas (es decir, que puedan apuntar a cualquier modelo).

Una instancia de *ContentType* representa un modelo particular de una aplicación en concreto. Los campos que contiene son:

- `name` - Cadena, nombre del modelo en formato legible.
- `model` - Cadena, nombre original del modelo.
- `app_label` - Cadena, nombre de la aplicación a la que pertenece el módulo.

5.4. Diseño de la imagen corporativa

La **imagen corporativa** del proyecto comprende el diseño del **logotipo**, la elección de la paleta de **colores** y las **tipografías**, que se usan en el diseño de la web, emails y cualquier otro elemento visual.

Para su elaboración se han seguido ciertas premisas o *guías de branding*, de forma que se consiguiese el look más adecuado:

- **Diseño minimalista.** En SiteUp, la información es lo más importante, así que no tiene sentido utilizar un diseño muy sobrecargado.
- **Colores monocromáticos.** Siguiendo en la línea del punto anterior, esta *regla* facilita que haya una consistencia en las interfaces de la web y la aplicación móvil sin tener que recurrir a complejas combinaciones de color.
- **Tipografías libres.** Era importante no utilizar tipografías que requiriesen una licencia comercial para su uso.

5.4.1. Diseño del logotipo

Partiendo del nombre del proyecto se pensaron varios bocetos para el logotipo. La idea principal era jugar con las dos palabras que componen el nombre: *Site* y *Up*. Para la primera palabra, la metáfora de usar una ventana de un navegador o algo que representase un servidor era demasiado compleja, así que se decidió simplemente usar la letra S.

Por otro lado, para la segunda palabra estaba claro desde el principio que se usaría alguna metáfora de tipo flecha para representar la dirección *arriba*. Se probaron distintos tipos de flechas y al final se decidió por una doble flecha de tipo galón, con suficiente separación para que se viese en bajas resoluciones. El logotipo resultante se puede ver en la figura 5.6



Figura 5.6: Logotipo de SiteUp

5.4.2. Elección de la gama cromática

Teniendo en mente la idea apuntada anteriormente de que la gama de colores debería ser preferiblemente monocromática, se empezó a buscar un tono que funcionase con la temática del sitio. Se hizo una búsqueda de sitios web de servicios de soporte, y en la mayoría de ellos la gama cromática era siempre muy neutra, con tonos poco saturados y casi siempre fríos.



Figura 5.7: Detalle de la paleta de colores de SiteUp

Así, tras varias pruebas, se decidió optar por una gama de azules poco saturados, rozando el grisáceo. En la figura 5.7 se detalla la paleta de colores principal.

5.4.3. Elección de la tipografía

Siguiendo las directrices apuntadas previamente, se buscaba una tipografía sencilla y minimalista. Se probaron algunas alternativas comerciales, empezando, cómo no, por Helvética y pasando por Myriad Pro, Próxima Nova y algunas otras. Al final nos decantamos por utilizar **Open Sans** [34], una tipografía sans-serif limpia y moderna, de corte humanista, y desarrollada por Steve Matteson para Google.

Se trata de una tipografía limpia, optimizada para la legibilidad en toda clase de soportes, y por supuesto libre, bajo una licencia Apache 2.0.

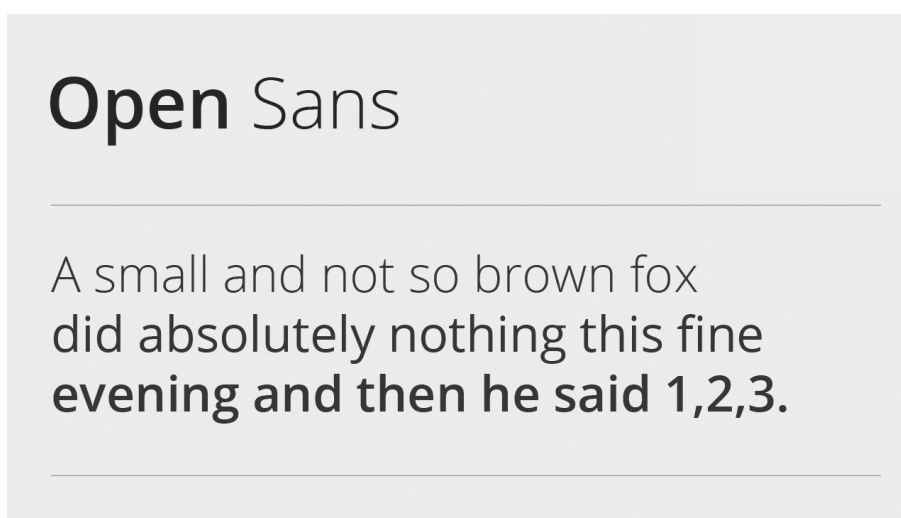


Figura 5.8: Muestra de Open Sans en diferentes pesos

5.5. Diseño de la interfaz de usuario de la plataforma web

En esta sección se detallarán las interfaces visuales de la plataforma web del proyecto, indicando qué acciones realizan cada uno de los componentes que las conforman.

5.5.1. Pantalla de inicio

En la figura 5.9 se ve el diseño de página de inicio, o *home*, de la plataforma web. Es la pantalla que aparece al acceder a la base del proyecto. Se componen de una barra superior de navegación en la que aparece el logotipo a la izquierda, y las opciones de navegación a la derecha. Al pulsar el logotipo, se redirige a la pantalla de inicio.

Si el usuario no está logueado, las opciones de navegación disponibles son:



Figura 5.9: Pantalla de inicio

- **Log in:** lleva al usuario a la pantalla de inicio de sesión.
- **Sign up:** lleva al usuario a la pantalla de registro de usuario.

Si el usuario sí está logueado, las opciones que aparecen son diferentes:

- **Dashboard:** lleva al usuario al listado de chequeos.
- **Your profile:** lleva al usuario al formulario de datos personales.
- **Log out:** cierra la sesión del usuario.

La barra de navegación superior tiene el mismo comportamiento en todas las secciones de la web.

5.5.2. Pantalla de inicio de sesión

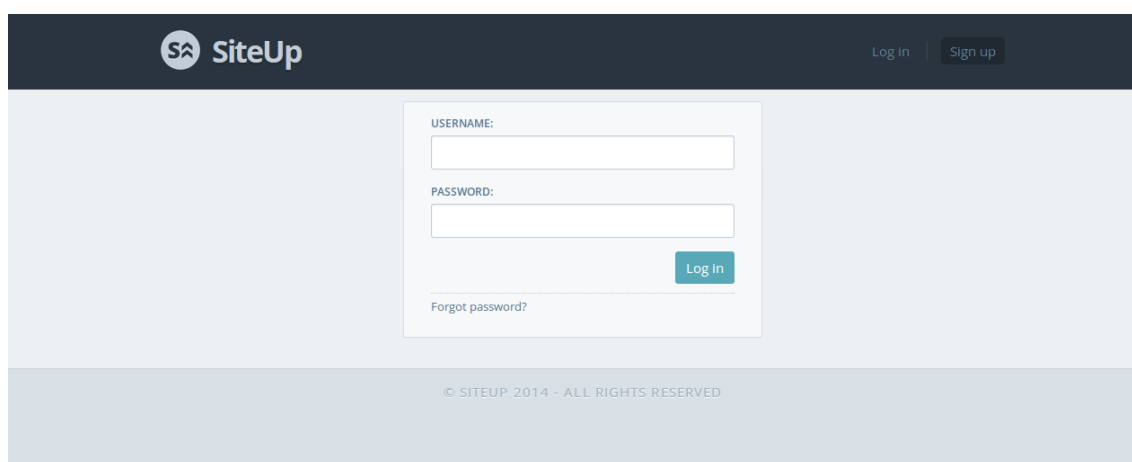


Figura 5.10: Pantalla de inicio de sesión

En la figura 5.10 se ve el diseño de la pantalla de inicio de sesión. Se presenta un formulario en el que el usuario deberá escribir su nombre de usuario y su contraseña, y tras ello pulsar el botón de *Login*. En caso de que los datos introducidos sean correctos, el sistema iniciará sesión y redirigirá al usuario a la lista de chequeos. En caso contrario, volverá a aparecer el formulario con los errores que se hayan producido.

Opcionalmente, si el usuario no recuerda su contraseña puede pulsar en el botón de *Forgot password?*, lo que le llevará a la pantalla de recuperación de contraseña.

5.5.3. Pantalla de recuperación de contraseña

En esta pantalla, visible en la figura 5.11, el usuario podrá introducir su correo electrónico para recuperar su contraseña. Tras rellenar el campo de texto, deberá pulsar el botón. Si el email introducido es correcto, el sistema enviará un correo

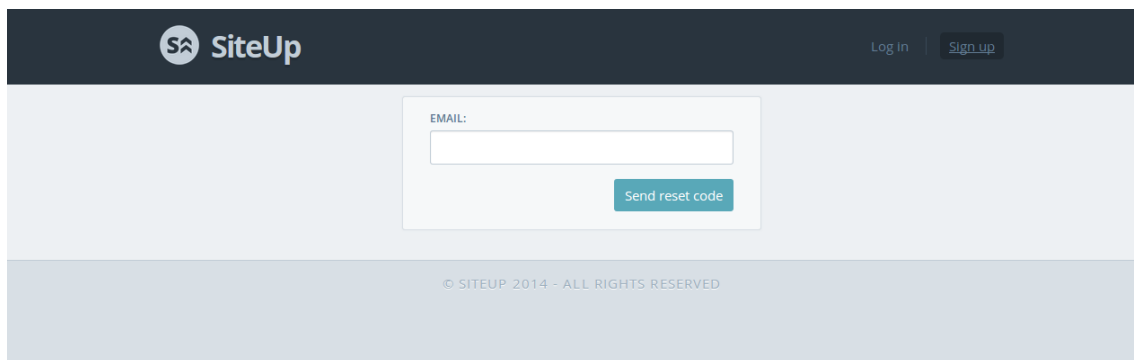
The screenshot shows the SiteUp password recovery interface. At the top, there is a dark header with the SiteUp logo on the left and 'Log in' and 'Sign up' links on the right. The main content area is light gray and contains a white-bordered box with an 'EMAIL:' label, a text input field, and a 'Send reset code' button. At the bottom of the page, there is a footer with the text '© SITEUP 2014 - ALL RIGHTS RESERVED'.

Figura 5.11: Pantalla de recuperación de contraseña

electrónico con instrucciones para reiniciar la contraseña, y en pantalla se mostrará un mensaje indicando que se ha iniciado el proceso de reinicio.

5.5.4. Pantalla de registro de usuario

En la figura 5.12 se puede ver la pantalla de registro de usuario.

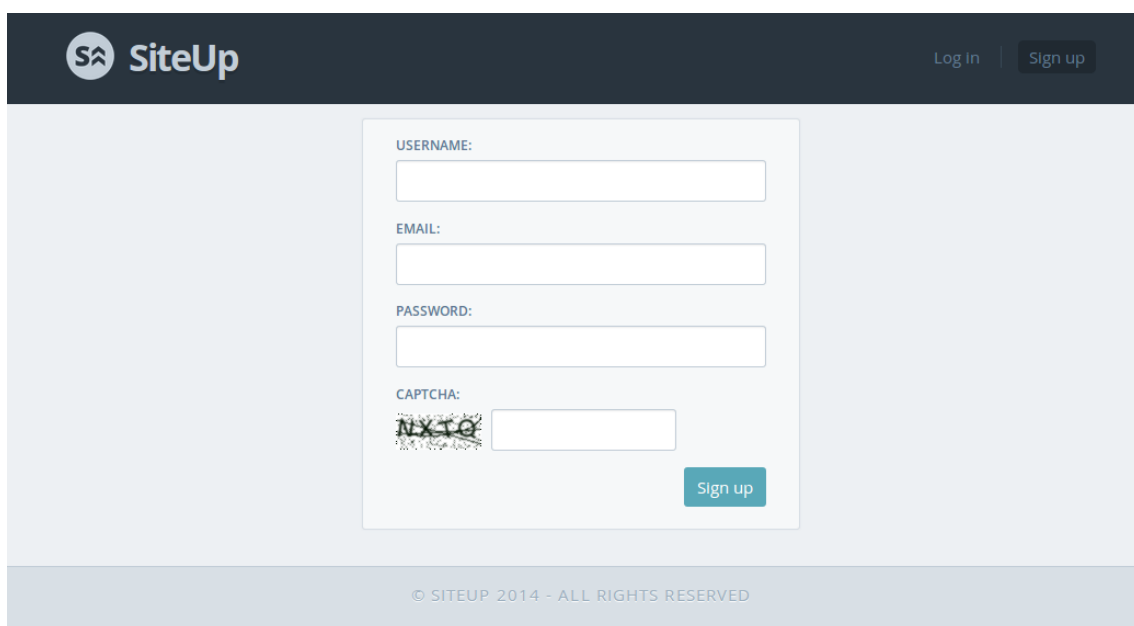
The screenshot shows the SiteUp user registration interface. It features a dark header with the SiteUp logo and 'Log in' and 'Sign up' links. The main content area is light gray and contains a white-bordered box with four input fields labeled 'USERNAME:', 'EMAIL:', 'PASSWORD:', and 'CAPTCHA:'. The CAPTCHA field includes a visual puzzle and a text input. A 'Sign up' button is located at the bottom right of the form box. The footer at the bottom of the page reads '© SITEUP 2014 - ALL RIGHTS RESERVED'.

Figura 5.12: Pantalla de registro de usuario

Dispone de un formulario, en el que el usuario deberá introducir sus datos personales: nombre de usuario, dirección de correo electrónico y contraseña. Una vez rellenos, deberá pulsar el botón. Si los datos son correctos, el sistema creará la cuenta de usuario. Si no, el sistema volverá a mostrar el formulario, resaltando los errores en los datos.

5.5.5. Pantalla de lista de chequeos

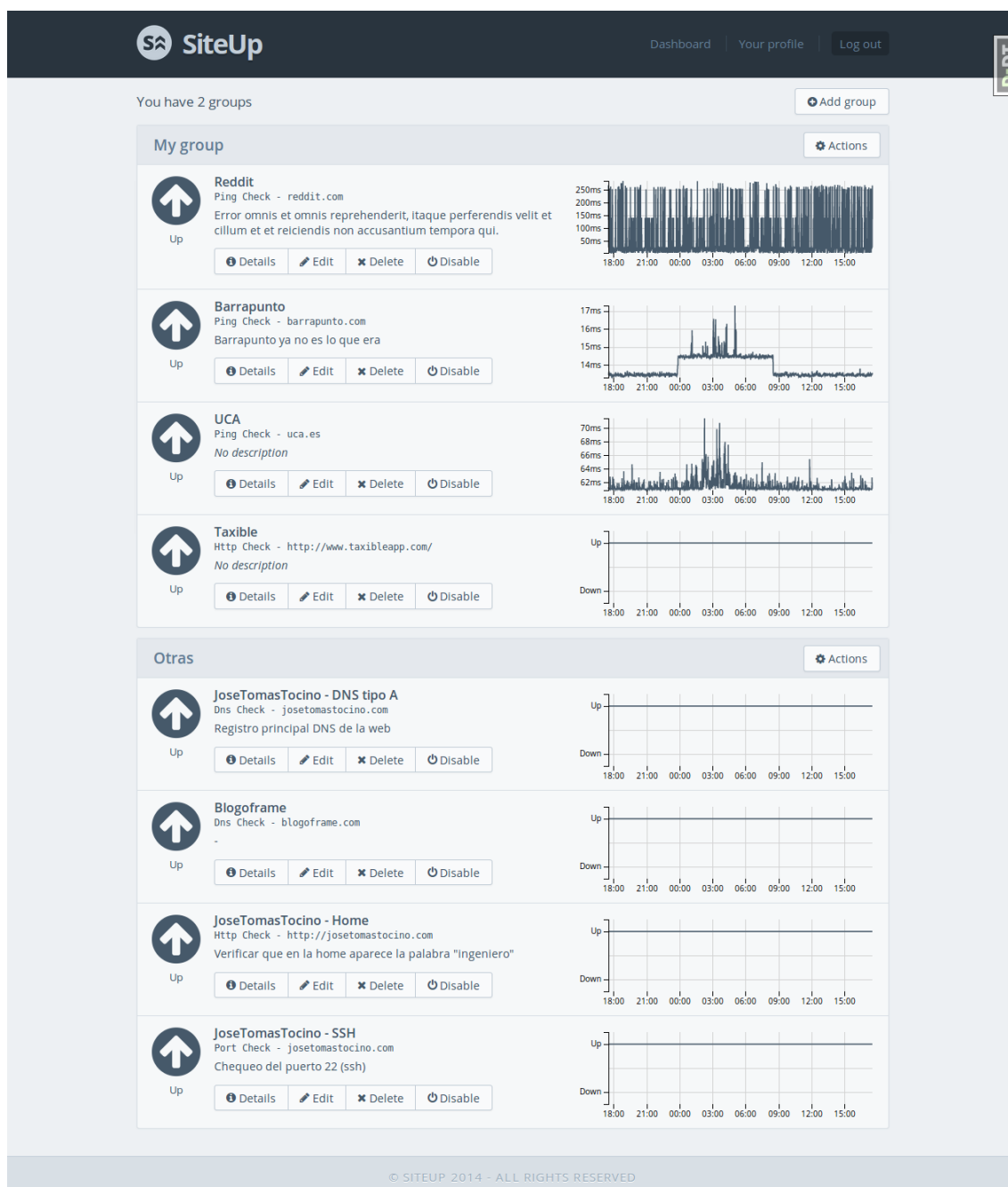


Figura 5.13: Pantalla de lista de chequeos

En la figura 5.13 se puede ver la pantalla principal de SiteUp, la lista de chequeos o **dashboard**. En esta pantalla aparecen los grupos de chequeos que tiene dados de alta un usuario, así como los chequeos que pertenecen a cada grupo. Desde esta pantalla se pueden hacer un gran número de operaciones:

- Pulsando el botón *Add group* aparecerá la pantalla de creación de grupo.
- Cada grupo tiene un menú desplegable que aparece al pulsar o colocar el

ratón sobre el botón *Actions*. Desde ahí, es posible:

- Añadir un chequeo al grupo pulsando el botón *Add check*.
 - Activar todos los chequeos del grupo pulsando el botón *Enable all checks*.
 - Desactivar todos los chequeos del grupo pulsando el botón *Disable all checks*.
 - Editar los detalles del grupo pulsando el botón *Edit*.
 - Borrar el grupo y los chequeos que contiene pulsando el botón *Delete*.
- Dentro de cada grupo aparece cada uno de los chequeos que contiene, con un icono indicando el estado del chequeo, el título, descripción, una gráfica de su estado en las últimas 24 horas y una serie de botones de acción, que permiten:
- Ver los detalles del chequeo pulsando el botón *Details*.
 - Editar las propiedades del chequeo con el botón *Edit*.
 - Borrar el chequeo con el botón *Delete*.
 - Activar o desactivar el chequeo pulsando *Enable* o *Disable*.

5.5.6. Pantalla de detalles de chequeo

En la figura 5.14 se puede ver la pantalla de detalle de un chequeo, a la que se accede pulsando el botón *details* desde el panel de control principal. Desde esta pantalla es posible realizar operaciones con el chequeo (editar, borrar y desactivar), así como ver información y eventos ocurridos en distintos intervalos de tiempo.

5.5.7. Pantalla de creación de grupo

La figura 5.15 representa la pantalla para crear un grupo de chequeos. Muestra un formulario en el que hay que introducir los detalles del grupo, concretamente el título que lo describe.

Cuando el usuario introduzca el título y pulse el botón, el sistema revisará que los datos sean correctos. En caso afirmativo, el sistema creará el nuevo grupo. En caso negativo, se volverá a mostrar el formulario, indicando los fallos ocurridos.

5.5.8. Pantallas de creación de chequeo

La creación de un chequeo se divide en dos pantallas principalmente. El primer paso es elegir el tipo de chequeo a crear, mediante los botones que aparecen en la pantalla de la figura 5.16.



Figura 5.14: Pantalla de detalles de chequeo

The screenshot shows the SiteUp dashboard with the 'Create group of checks' form. The form has a 'TITLE:' label and a text input field. Below the input field is a 'Create group' button. The footer of the dashboard shows '© SITEUP 2014 - ALL RIGHTS RESERVED'.

Figura 5.15: Pantalla de creación de grupo de chequeos

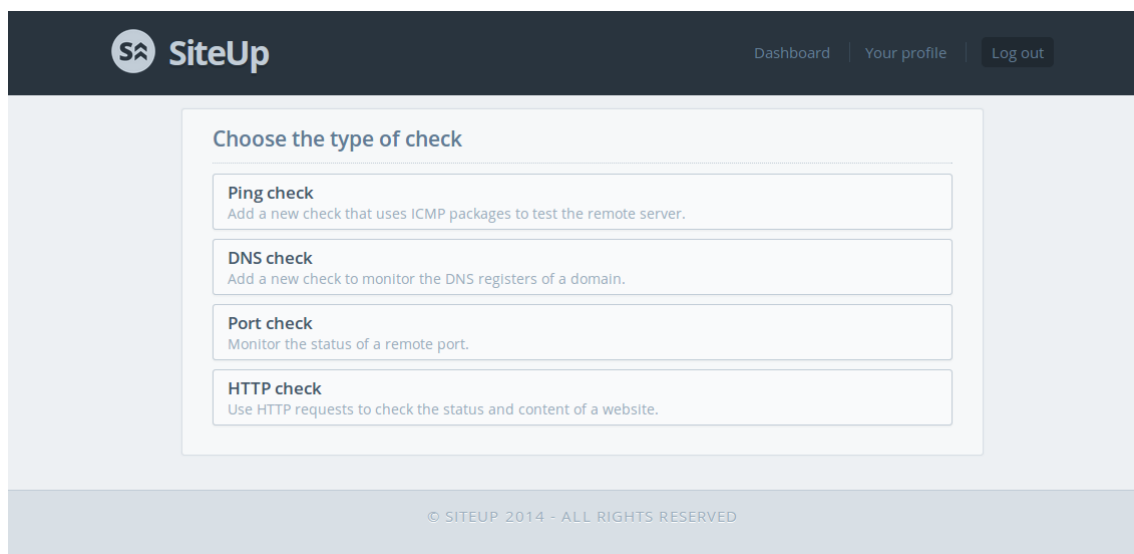


Figura 5.16: Pantalla de elección de tipo de chequeo

Una vez elegido el tipo de chequeo a crear, aparecerá un formulario para introducir los detalles del chequeo, que variará según el tipo de chequeo elegido. En la figura 5.17 se puede ver el formulario que aparecerá a la hora de crear un chequeo de tipo DNS.

Cuando el usuario rellene el formulario y pulse el botón, se enviarán los datos y el sistema comprobará que la información es correcta. En tal caso, se creará el nuevo chequeo y se redirigirá al usuario al *dashboard*. En caso contrario, se volverá a mostrar el formulario, indicando los errores encontrados.

5.5.9. Pantalla genérica de borrado

En la figura 5.18 se representa la pantalla de confirmación de borrado que aparece cuando se intenta borrar un chequeo o un grupo de chequeos. La pantalla es sencilla: pregunta al usuario si desea realmente borrar el elemento, ofreciendo dos botones, uno para confirmar el borrado y otro para cancelar la operación. En ambos casos, tras la interacción el sistema redirige al usuario al *dashboard*.

5.5.10. Pantalla del perfil de usuario

En la figura 5.19 se muestra la pantalla para editar el perfil de usuario, a la que se llega pulsando el botón *Your profile* situado en la barra superior de navegación.

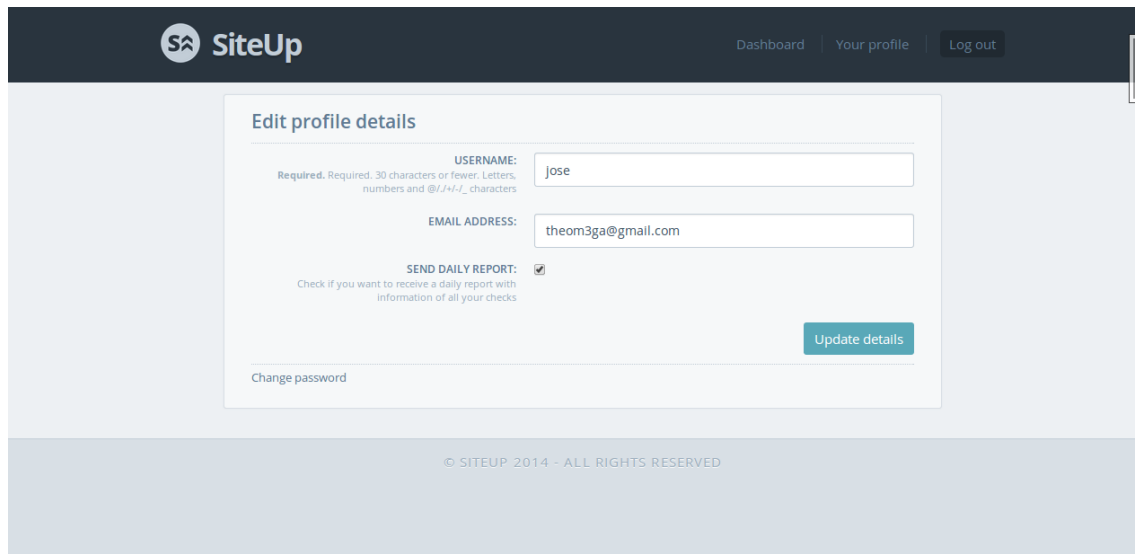
Desde esta pantalla el usuario puede modificar sus datos de usuario: nombre de usuario, correo electrónico y si desea recibir un reporte diario sobre el estado de sus chequeos.

The screenshot shows the 'Create new Dns check' form in the SiteUp web application. The form is titled 'Create new Dns check' and is located in the center of the page. It contains several input fields and checkboxes. The fields are: 'TITLE' (Required, Short title for the check), 'DESCRIPTION' (Required, Larger description, you can include notes), 'TARGET' (Required, Should be a hostname), 'RECORD TYPE' (Required, Type of dns resource record, with a dropdown menu showing 'A'), 'RESOLVED ADDRESS' (Required, Should be a valid value for the selected record type), 'CHECK INTERVAL' (Required, In minutes, How often the check should be triggered, with a slider set to 1), and 'CONSECUTIVE LOGS FOR FAILURE' (Required, Number of consecutive checks to consider a status change, with a slider set to 2). There are also two checkboxes: 'NOTIFY EMAIL' (Notify changes of the status of this check via email) and 'NOTIFY ANDROID' (Notify changes of the status of this check through the Android app (if available)). A 'Create check' button is located at the bottom right of the form. The SiteUp logo and navigation links (Dashboard, Your profile, Log out) are visible in the top header. The footer contains the copyright notice: © SITEUP 2014 - ALL RIGHTS RESERVED.

Figura 5.17: Pantalla de introducción de datos del chequeo

The screenshot shows the 'Confirm deletion' dialog in the SiteUp web application. The dialog is titled 'Confirm deletion' and is located in the center of the page. It contains the text 'Are you sure you want to delete the check?' and 'Title: Reddit'. There are two buttons: 'Go back' and 'Delete'. The SiteUp logo and navigation links (Dashboard, Your profile, Log out) are visible in the top header. The footer contains the copyright notice: © SITEUP 2014 - ALL RIGHTS RESERVED.

Figura 5.18: Pantalla genérica de borrado



The screenshot shows the 'Edit profile details' page of the SiteUp application. The header is dark blue with the SiteUp logo on the left and navigation links 'Dashboard', 'Your profile', and 'Log out' on the right. The main content area is light gray and contains a white form titled 'Edit profile details'. The form has three input fields: 'USERNAME:' with the value 'jose', 'EMAIL ADDRESS:' with the value 'theom3ga@gmail.com', and a checkbox for 'SEND DAILY REPORT:' which is checked. Below the form is a blue 'Update details' button and a link 'Change password'. The footer is a dark gray bar with the text '© SITEUP 2014 - ALL RIGHTS RESERVED'.

SiteUp

Dashboard | Your profile | Log out

Edit profile details

USERNAME:

Required. Required. 30 characters or fewer. Letters, numbers and @/./+/-/_ characters

EMAIL ADDRESS:

SEND DAILY REPORT: ☒

Check if you want to receive a daily report with information of all your checks

[Update details](#)

[Change password](#)

© SITEUP 2014 - ALL RIGHTS RESERVED

Figura 5.19: Pantalla del perfil de usuario

También cuenta con la opción de cambiar la contraseña pulsando el botón *Change password*.

5.6. Diseño de la interfaz de usuario de la aplicación móvil

En esta sección se detallarán las interfaces visuales de la aplicación móvil para sistemas operativos Android, indicando la navegabilidad e interacción entre las pantallas.

5.6.1. Aspecto desde el lanzador

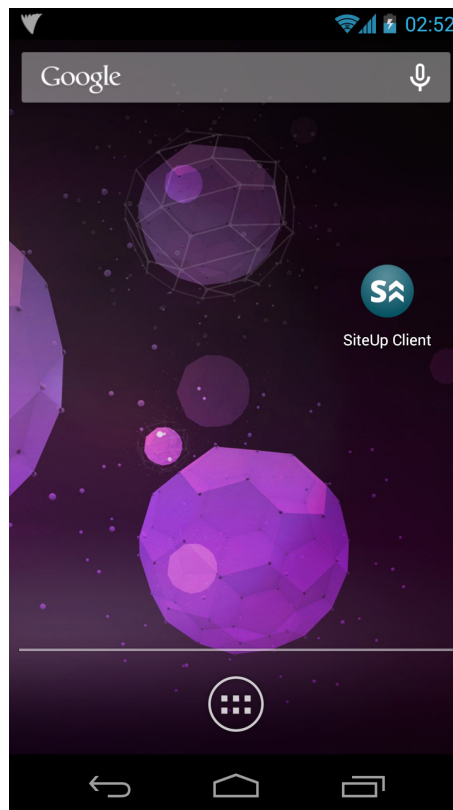


Figura 5.20: Icono de SiteUp en el lanzador

La figura 5.20 muestra el aspecto del icono de la aplicación desde el lanzador de aplicaciones de Android.

Como se comentó en la sección 5.4.1, el logotipo fue diseñado pensando en su legibilidad en bajas resoluciones, por lo que el icono de la aplicación es fácilmente identificable desde el lanzador.

Al pulsar el icono de la aplicación, se lanza la pantalla de carga.



Figura 5.21: Pantalla de carga de la aplicación móvil

5.6.2. Pantalla de carga

En cuanto se pulsa el lanzador aparece la pantalla de carga de la aplicación, visible en la figura 5.21. Esta pantalla permanece visible poco tiempo ya que el tiempo de carga es rápido.

Tras el proceso de carga, si la aplicación Android detecta que el usuario ha iniciado sesión previamente, se carga la pantalla con el listado de chequeos. En caso de que no haya datos de una sesión anterior, se carga la pantalla de login.

5.6.3. Pantalla de inicio de sesión

Tras cargar la aplicación, si no se detectan datos de inicio de sesión previos se muestra la pantalla de inicio de sesión, visible en la figura 5.22, en la que el usuario debe introducir su nombre de usuario y contraseña para iniciar sesión.

5.6.4. Pantalla de listado de chequeos

La figura 5.23 muestra la pantalla con el listado de chequeos del usuario. Desde aquí es posible cerrar la sesión de usuario, o pulsar en alguno de los chequeos para

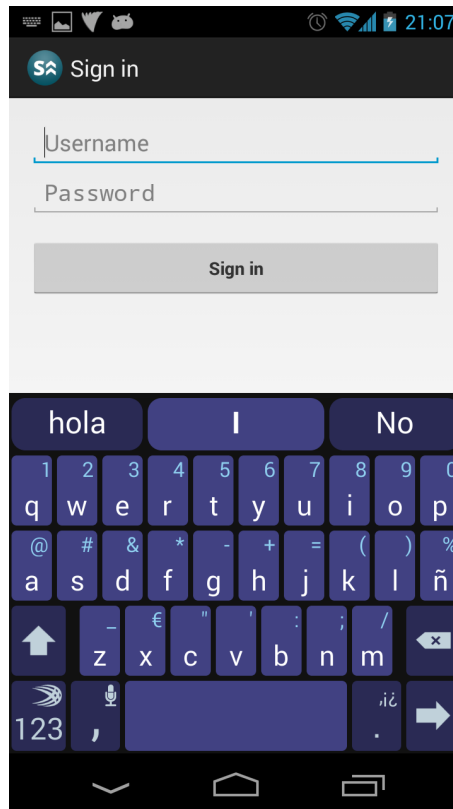


Figura 5.22: Pantalla de inicio de sesión de la aplicación móvil

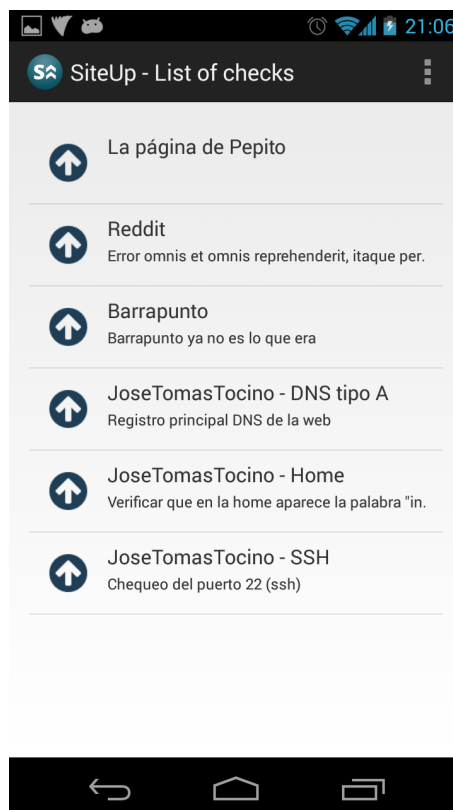


Figura 5.23: Pantalla de chequeos de la aplicación móvil

ver sus detalles, lo cual abrirá un navegador y mostrará la plataforma web.

5.6.5. Notificación

La figura 5.24 muestra el aspecto que tienen las notificaciones que la aplicación es capaz de recibir. Desde estas notificaciones es posible dirigirse a la web para ver los detalles del chequeo involucrado.

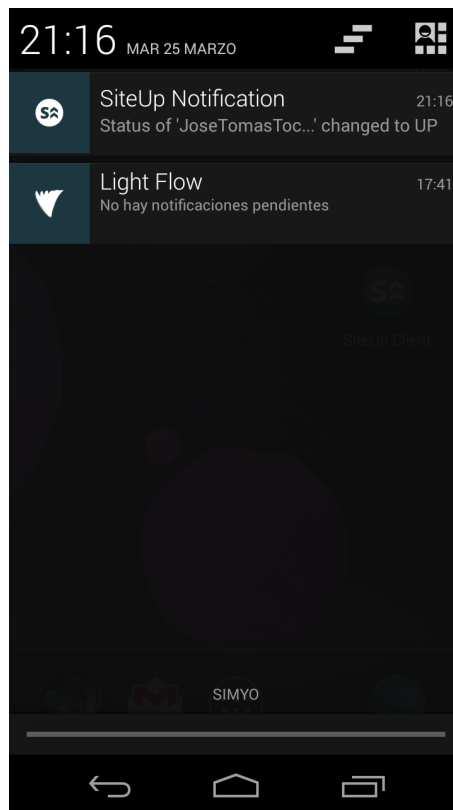


Figura 5.24: Aspecto de las notificaciones

Capítulo 6

Implementación

Como complemento a la lectura de este capítulo se recomienda tener una copia local del repositorio del proyecto, disponible para su libre descarga desde la forja oficial [27].

6.1. Entorno de construcción

En esta sección se detalla el marco tecnológico utilizado para la construcción del sistema, basándonos en lo presentado en la sección 3.3, *Alternativas de solución*.

6.1.1. Aplicación web

El entorno tecnológico de la aplicación web ha resultado ser muy variado y rico en tecnologías en total vigencia en la actualidad.

Herramientas de diseño y desarrollo

Como editor principal se ha utilizado **Sublime Text 2** [42], un editor freeware escrito en Python especialmente dotado para el desarrollo de aplicaciones web. Cuenta con un gran número de características que lo han hecho destacar en los últimos años, como la habilidad de seleccionar y editar a la vez varios fragmentos de texto, el rápido motor de *búsqueda difusa* para encontrar ficheros y cadenas o su capacidad de extensión a través de paquetes escritos por terceros y fácilmente instalables.

Para ciertas tareas, como la escritura de la presente memoria, se ha hecho uso del veterano editor **Emacs** [20], que sigue siendo la mejor opción a la hora de editar documentos de \LaTeX .

Se ha utilizado **draw.io** [15] para la generación de diagramas de casos de uso, secuencia, diagramas de flujo y similares. Se trata de una herramienta web, actualmente integrable en Google Drive, con un enorme número de elementos de dibujo y la capacidad de exportar en un gran número de formatos, entre ellos en formato vectorial PDF (Portable Document Format). Para el diseño visual de las interfaces de usuario se ha utilizado **Adobe Photoshop** [1].

Gestión de dependencias

Para facilitar la gestión de las dependencias del proyecto se han utilizado **VirtualEnv** [44] y **VirtualEnvWrapper** [45]. Estas herramientas permiten generar *entornos virtuales* para cada proyecto, en los que se instalan las dependencias necesarias. Estos entornos se activan y desactivan, de forma que las bibliotecas instaladas en el entorno virtual de un proyecto no son accesibles desde el entorno del otro proyecto. Esto evita la polución del nivel general de bibliotecas y facilita el control estricto de las versiones de las dependencias.

Junto a **virtualenv**, la herramienta **pip** [36] permite guardar en un fichero anexo la lista de dependencias de un proyecto, de forma que sea fácil reinstalarlas todas si hubiese que repetir la instalación en otro sistema.

Control de versiones

Todo el código fuente del proyecto se encuentra alojado en un repositorio público en **GitHub** [27], haciendo uso de los planes gratuitos. GitHub es una forja que utiliza el sistema de control de versiones **Git** [19]. Además del alojamiento de código, GitHub provee numerosas funcionalidades adicionales, tanto a nivel social (permitiendo a las forjas tener *followers*, por ejemplo) como a nivel funcional (ofreciendo sistemas de tickets, estadísticas, etcétera).

El uso de un control de versiones es fundamental por varios motivos. En primer lugar, sirve como sistema de copia de seguridad. En segundo lugar, permite deshacer cambios en el código que no funcionen bien, siendo siempre posible *volver atrás*. Por último, sirve como *cuaderno de bitácora* improvisado, ya que se guarda el historial de *commits* que el desarrollador va enviando junto a los mensajes, siendo posible ver en una línea temporal el progreso del trabajo.

Lenguaje de programación

Como se ha comentado en numerosas ocasiones en la presente memoria, el lenguaje de programación elegido para el desarrollo del proyecto es **Python** [37], un lenguaje interpretado de alto nivel desarrollado por Guido Van Rossum en 1991. Python soporta múltiples paradigmas de programación, desde la orientación a objetos hasta la programación funcional, pasando por el clásico estilo imperativo. Su

principal uso ha sido como lenguaje de scripting, pero también tiene su hueco en contextos más amplios como lenguaje principal.

Su facilidad de aprendizaje, lo simple de su sintaxis (basada en la indentación para marcar los bloques) y su extensibilidad (sobre todo gracias al uso de métodos *mágicos* y metaprogramación) han hecho que el lenguaje sea muy popular y su uso en los últimos años se haya expandido enormemente.

Nivel de persistencia

SQLite es un sistema de gestión de bases de datos relacional de dominio público, contenido en una pequeña biblioteca. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos.

El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción. Esto no impide que varios procesos o hilos puedan acceder a la misma base de datos sin problemas.

En general, SQLite no suele ser la mejor opción en sistemas de producción muy extensos, pero para proyectos de menor envergadura su rendimiento es más que suficiente.

Herramientas de frontend

Para el desarrollo del frontend de la web, esto es, lo que el usuario acaba viendo y utilizando en el navegador, se han utilizado varias tecnologías, herramientas y bibliotecas.

En primer lugar, la base de la web está escrita en **HTML5** [25], un lenguaje de marcado usado para estructurar y presentar contenido en la web. Es la quinta revisión del estándar HTML (Hyper Text Markup Language), y su objetivo principal es unificar la sintaxis tan dispersa que había generado el uso de HTML4 y XHTML. Además, introduce una serie de elementos (etiquetas) con contenido semántico, tales como `<section>` y `<article>`. En la actualidad la especificación de HTML5 no está cerrada y siguen añadiéndose cambios. Es por ello que la compatibilidad con los navegadores no es total, aunque sí muy extensa.

Una buena práctica en el desarrollo web es desacoplar lo máximo posible el contenido de un documento (definido con HTML) del diseño del mismo, que habitualmente se define usando **CSS** [8]. En particular, en SiteUp se ha utilizado CSS3, la

última versión del estándar. CSS permite la definición de **estilos** que se aplican a **elementos** de una web, mediante **reglas**. Así, las reglas se componen de un selector, que define los elementos a los que se le aplicará el estilo, y un bloque de declaración, en el que se describen las propiedades que se le aplicarán al elemento. Hay un gran número de reglas para la definición de selectores, y una extensa lista de propiedades definibles en CSS.

A pesar de todo, CSS cuenta con una serie de limitaciones de base. Por ejemplo, no cuenta con mecanismos sencillos para favorecer la modularidad del código. Por ello han surgido una serie de *metalenguajes* que añaden nuevas características para facilitar el desarrollo pero que, al final, acaban generando código CSS compatible. Uno de estos metalenguajes es **Sass (Syntactically Awesome Style Sheets)** [38]. Entre las funcionalidades que Sass añade se encuentran:

- Definición de variables.
- Anidamiento de reglas.
- Uso de *mixins*.
- Herencia de selectores.

Además, Sass cuenta con bibliotecas de terceros, como **Compass** [10], un framework de desarrollo que añade a Sass numerosas utilidades adicionales.

Con HTML5 y CSS3 es posible crear perfectamente sitios web con animaciones y funcionalidad básica, pero en SiteUp era necesario tener cierta interactividad a la hora de, por ejemplo, mostrar gráficas. Para ello resulta necesario utilizar un lenguaje de scripting para web, y ese lenguaje es **JavaScript** [54]. Se trata de un lenguaje dinámico utilizado principalmente para añadir dinamismo a los sitios web. Se lanzó en 1995 y desde entonces ha ido mejorando con el tiempo, sobre todo con la adición de numerosas API (Application Programming Interface) en los navegadores. El problema que ha tenido JavaScript es el irregular soporte de los diferentes navegadores, sobre todo las versiones de Internet Explorer. Así, era habitual utilizar código distinto para realizar el mismo procedimiento según fuese el navegador.

Afortunadamente existen bibliotecas de desarrollo que actúan como capa de abstracción y permiten al desarrollador olvidarse de las diferencias de compatibilidad entre navegadores. La biblioteca más conocida para uso general es **jQuery** [28]. Creada en 2006 por John Resig, se estima que más del 50 % de **todos** los sitios web a nivel mundial usan jQuery. Su popularidad se debe sobre todo a su facilidad de uso, su ligereza y la potencia que ofrece. Por suerte la evolución de los navegadores está permitiendo que el soporte nativo de JavaScript sea cada vez mejor, siendo progresivamente menos necesario el uso de bibliotecas *unificadoras* como jQuery, aunque su utilidad a la hora de trabajar, por ejemplo, con efectos o con cargas asíncronas de datos AJAX (Asynchronous JavaScript and XML) sigue siendo vigente.

Para el renderizado y muestra de las gráficas, en SiteUp se ha usado **D3.js** [12], una biblioteca para la manipulación de documentos basada en datos. D3 permite

generar interesantes visualizaciones de multitud de tipos, teniendo además capacidad para añadir interactividad, animaciones y propiedades dinámicas. Las gráficas generadas por D3 normalmente utilizan SVG (Structured Vector Graphics), por lo que el rendimiento es muy alto.

Bibliotecas de terceros para backend

En el proyecto se han utilizado un gran número de bibliotecas auxiliares para facilitar el desarrollo y ampliar la funcionalidad de forma sencilla.

South South [39] es una biblioteca para Django que permite generar **migraciones** de datos y estructurales de la base de datos. El término *migración* hace referencia a una modificación estructural en la que se definen de forma lógica las diferencias entre el estado anterior y el posterior, de forma que sea posible trasladar o *transformar* los datos de un estado a otro, incluso dando la posibilidad de deshacer los cambios.

Es una función de gran utilidad, y muchos otros frameworks, como *Ruby on Rails*, incorporan este mecanismo de forma nativa. South se ha hecho tan popular y común que su código ha sido integrado en Django de forma nativa a partir de la versión 1.7.

Fabric Fabric [16] se compone de una herramienta de línea de comandos y una biblioteca que facilitan la comunicación con servidores remotos a través de SSH, facilitando el despliegue de aplicaciones y las tareas de administración.

Provee un conjunto básico de operaciones para ejecutar, ya sea forma local o remota, operaciones de toda clase: desde comandos de shell hasta subida y descarga de ficheros, pasando por funciones auxiliares.

En SiteUp, Fabric se ha utilizado para facilitar el despliegue de la plataforma web en el servidor remoto. Para ello, se desarrolló un fichero `fabfile.py`, que se encuentra en la raíz del proyecto web, que define las operaciones a realizar. Cuando se ejecuta el siguiente comando:

```
$ fab deploy
```

El sistema realiza las siguientes operaciones en el servidor remoto:

- Activa el entorno virtual (*virtualenv*).
- Descarga los últimos cambios del código desde la forja [27].
- Instala los nuevos requisitos definidos mediante el fichero de requisitos del entorno virtual (véase la sección 6.1.1 *Gestión de dependencias*).
- Dispone los ficheros estáticos del proyecto en un directorio especial, para que el servidor web pueda servirlos (véase la sección 5.1.2, *Plataforma web*).

- Aplica cualquier cambio que se haya realizado a la estructura de la web mediante migraciones.
- Reinicia los servicios.

La reducción de tiempo es considerable y la facilidad de uso es total.

django-braces Se trata de una extensión de Django que provee un conjunto de *mixins* que agregan funcionalidad a las vistas de forma sencilla.

Un **mixin** es una clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero que no está pensada para funcionar por sí misma de forma autónoma. Así, la herencia que se produce al heredar de un mixin es básicamente un mecanismo para integrar funcionalidad adicional, en lugar de una herencia en el sentido clásico.

Entre la funcionalidad que provee django-braces se encuentran, por ejemplo, mixins para controlar el acceso a zonas para usuarios registrados o para devolver fácilmente respuestas JSON.

django-extensions Se trata de una colección de extensiones para Django de muy diversos tipos: comandos de administración, extensiones para la zona de administración, etcétera.

Algunas de las utilidades más interesantes que provee son las siguientes:

- El comando `shell_plus` provee una shell que carga automáticamente todos los modelos de la aplicación, de forma que sea fácil hacer consultas rápidas a la base de datos.
- El comando `graph_models` genera automáticamente un diagrama visual con el modelo Entidad-Relación de la base de datos.

django-debug-toolbar Proporciona una barra con extensa información de depuración: tiempos de cpu y renderizado, consultas SQL ejecutadas, plantillas utilizadas, variables de entorno y de petición, ficheros estáticos, señales dadas de alta, mensajes de registro...

Esta extensión es uno de los mecanismos de depuración más utilizados en los proyectos Django de todo el mundo. Además, automáticamente se desactiva en entornos de producción, por lo que no supone un problema.

6.1.2. Aplicación móvil

El entorno tecnológico de la aplicación móvil es el estándar para el desarrollo de aplicaciones Android hoy en día, que consta de los siguientes elementos.

Herramientas de desarrollo

Para el desarrollo de la aplicación móvil se ha optado por utilizar **Android Studio** [4], un entorno de desarrollo integrado basado en el popular software IntelliJ IDEA y perfectamente personalizado con las bibliotecas y herramientas necesarias para el desarrollo de aplicaciones Android.

Está preparado y gestionado por Google, y aunque está en una etapa temprana de desarrollo, su uso es totalmente viable y los resultados son correctos. Incluye elementos muy útiles, como la integración con los servicios de **Google Cloud Messaging**, que se han utilizado en el proyecto (véase la sección 5.1.2, *Aplicación Android*).

Servicio de notificaciones

Para el envío y la recepción de notificaciones se ha utilizado el servicio **Google Cloud Messaging for Android**. Se trata de un servicio gratuito que permite enviar datos desde un servidor a dispositivos Android que previamente se hayan registrado a través de una aplicación en particular.

El flujo de funcionamiento es el siguiente:

1. El usuario instala la aplicación, por ejemplo *SiteUp Client*.
2. La aplicación conecta con los servidores de GCM y pide el **código de registro** del dispositivo, que lo identifica unequivocamente.
3. La aplicación guarda ese código en el servidor de la plataforma web.
4. En caso de necesitar enviar una notificación, la plataforma web hace una petición al servidor de GCM, enviando el código de registro del dispositivo al que enviar la notificación y, lógicamente, el contenido de la notificación.
5. El servidor de GCM se encarga de transmitir la notificación al dispositivo Android.
6. El dispositivo recibe la notificación, siendo la aplicación móvil la encargada de actuar en consecuencia, mostrando un mensaje o realizando cualquier otra operación.

La facilidad con la que se pueden enviar mensajes de esta manera, de forma instantánea, es asombrosa y pone de manifiesto las facilidades que la plataforma Android pone a los desarrolladores.

6.2. Organización del código fuente

En esta sección se detalla la organización del código fuente del proyecto, describiendo la utilidad de los ficheros y directorios.

El código fuente presente en la forja [27] cuenta con los siguientes elementos en el directorio raíz:

- `android-client`: código fuente de la aplicación Android.
- `logs`: directorio con los ficheros de registro.
- `memoria`: directorio con el código fuente de la presente memoria.
- `web`: código fuente de la plataforma web.
- `LICENSE`: fichero con el texto de la licencia.
- `INSTALLING-web.md`: fichero con instrucciones de instalación de la plataforma web.
- `README.md`: fichero de presentación para la forja.

6.2.1. Plataforma web

El código de la plataforma web se encuentra íntegramente contenido en el directorio `web` del proyecto, que contiene lo siguiente:

- `db`: directorio donde se guarda la base de datos y otros ficheros.
 - `celerybeat.db`: datos del servicio Beat de Celery.
 - `django-db.sqlite3`: base de datos principal SQLite.
- `siteup`: directorio con ficheros a nivel de proyecto.
 - `settings`: directorio de configuraciones.
 - `base.py`: configuración general.
 - `local.py`: configuración para entorno local de desarrollo.
 - `production.py`: configuración para entorno de producción
 - `celery.py`: configuración de Celery.
 - `urls.py`: configuración de rutas de acceso web y URLs.
 - `wsgi.py`: configuración del punto de acceso para el servidor WSGI.
- `siteup_api`: directorio de la app que gestiona la API.
 - `migrations`: directorio de migraciones generadas por South.
 - `tests`: directorio con los tests de la aplicación.
 - `admin.py`: configuración de la app para el panel de administración.
 - `managers.py`: gestores de modelos personalizados.
 - `models.py`: definiciones de los modelos de la base de datos.
 - `utils.py`: funciones auxiliares.

- `validators.py`: validadores personalizados para formularios y campos de modelos.
- `views.py`: vistas de acceso.
- `siteup_checker`: directorio de la app que gestiona las tareas de chequeo a bajo nivel.
 - `deployment`: ficheros auxiliares para las tareas de despliegue.
 - `management`: tareas personalizadas, a ser lanzadas mediante la línea de comandos.
 - `monitoring`: procedimientos para chequeos:
 - `ping.py`: chequeo mediante ping.
 - `http.py`: chequeo mediante peticiones HTTP.
 - `pport.py`: chequeo de puertos.
 - `ddns.py`: chequeo de registros DNS.
 - `templates`: plantillas para los correos de notificación.
 - `tests`: tests para el código de la aplicación.
 - `admin.py`: configuración de la app para el panel de administración.
 - `models.py`: modelos de la aplicación.
 - `tasks.py`: definiciones de las tareas de Celery.
 - `tasks_notification.py`: definiciones de las tareas de Celery relacionadas con la emisión de notificaciones.
- `siteup_frontend`: directorio de la app que gestiona el frontend de la web.
 - `static_src`: código fuente de los ficheros de los assets del front-end.
 - `js`: ficheros JavaScript.
 - `scss`: ficheros SASS.
 - `static`: código generado a partir de los fuentes en `static_src`. El flujo de trabajo del front-end se detalla en la sección 6.3.3.
 - `templates`: plantillas HTML para las páginas de la web.
 - `templatetags`: funciones personalizadas para usar en el código de las plantillas.
 - `context_processors.py`: ficheros de procesamiento del contexto personalizados.
 - `forms.py`: formularios adaptados para las secciones de la web, con código de validación.
 - `tests.y`: procedimientos de test para la aplicación.

- `views.py`: definición de las vistas (funciones que responden a las peticiones) de la web.
- `static`: directorio donde se agrupan los ficheros estáticos de la web.
- `Gruntfile.js`: fichero para el gestor de tareas Grunt.
- `package.json`: fichero de dependencias de `node.js`.
- `fabfile.py`: fichero de tareas de Fabric.
- `manage.py`: utilidad de línea de comandos de Django.
- `requirements.txt`: fichero de dependencias para el entorno virtual.

6.2.2. Aplicación móvil

Los ficheros de la aplicación móvil se encuentran contenidos en el directorio `android-client`. El contenido es el siguiente:

- `app`: directorio con el código fuente.
 - `build`: ficheros binarios compilados a partir del código fuente, entre los que se encuentra el fichero de compilación con extensión `.apk` que se instala en el dispositivo.
 - `libs`: bibliotecas de terceros.
 - `src`: código fuente. La ruta completa hasta los ficheros es `src/main/java/com/josetomastocino/siteupclient/app/`, siguiendo las prácticas habituales de los proyectos Java.
 - `CheckInList.java`: representa un chequeo en la lista de chequeos.
 - `CheckListActivity.java`: actividad que muestra la lista de chequeos.
 - `CheckListAdapter.java`: adaptador de array para la lista de chequeos.
 - `GcmBroadcastReceiver.java`: receptor de notificaciones GCM. Notifica al servicio `GcmIntentService`.
 - `GcmIntentService.java`: manejador de notificaciones GCM.
 - `LoginActivity.java`: actividad que gestiona el inicio de sesión.
 - `SplashScreenActivity.java`: actividad que gestiona la carga y el registro del dispositivo en Google Play Services.
- `gradle*`: ficheros de configuración para el motor de compilación Gradle.

6.3. Detalles de implementación

En esta sección se describen algunos detalles en la implementación del proyecto que han resultado de interés suficiente para ser dignos de mención, ya sea por los conflictos que dieron, por su complejidad o por otras razones subjetivas.

6.3.1. Implementación de los chequeos

El núcleo del proyecto, como se ha comentado en varias ocasiones, es el módulo `siteup_checker`, que es el encargado de llevar a cabo los chequeos contra los servicios web. La implementación de estos chequeos no estuvo exenta de dificultades y *corner cases* que se fueron depurando progresivamente.

Implementación del chequeo por ping

Uno de los primeros problemas que se encontraron fue que para poder enviar paquetes en el protocolo ICMP es necesario tener permisos de administrador (o *root*) en el sistema, cosa que no es habitual ni recomendable al ejecutar una aplicación web. Este requisito se da porque es necesario abrir un socket puro (*raw socket*), un tipo especial de socket que solo se usa en ocasiones especiales y cuya creación se restringe al administrador del sistema [29].

La alternativa más habitual es ejecutar el comando **ping** de forma externa, y procesar su salida. El comando `ping` tiene un permiso de acceso especial que permite a cualquier usuario lanzar el ejecutable emulando temporalmente los privilegios del usuario `root`.

Para procesar la salida del comando se utilizó una expresión regular de gran envergadura, que tenía en cuenta casos especiales como aquellos en los que los paquetes ICMP se perdiesen. Se presenta a continuación un fragmento del código involucrado.

```
process = subprocess.Popen(["ping", "-c3", "-w2", target],
                           stdout=subprocess.PIPE,
                           stderr=subprocess.PIPE)

# Get the output
ping_raw_response, ping_raw_error = process.communicate()

# ...

# Compile regular expression to parse ping's output
matcher = re.compile(r"""
^PING \s+                # Header
(?P<host>.*?) \s+         # Host
```

```

\((?P<ip>.*?)\)          # IP address

.*?                      # Ignore individual pings

(?P<transmitted>\d+) \s+ packets      .*?      # Packets transmitted
(?P<received>\d+) \s+ received        .*?      # Packets received
ms                                   \s*      # Ending of the statistics line
(                                     # Non-capturing group for the stats
rtt .* = \s+                        # Separator
(?P<min>[^\s]*)/                    # Min time
(?P<avg>[^\s]*)/                    # Avg time
(?P<max>[^\s]*)/                    # Max time
(?P<mdev>.*?) \s ms                 # Mdev
)?

""", re.DOTALL | re.IGNORECASE | re.MULTILINE | re.VERBOSE)

# Run the regular expression to parse ping's output
results = matcher.match(ping_raw_response)

# If it doesn't match (due to an error)
if not results:
    logger.error("PING ERROR" + ping_raw_response)

# ...

```

Implementación del chequeo de DNS

Uno de los detalles que se encontraron en la codificación de este chequeo es que las peticiones a los servidores de nombres se hacen utilizando el tipo de registro como único criterio (obviamente además del dominio), por lo que el resultado es múltiple. Por ejemplo, una posible petición podría ser conocer los registros tipo A del dominio google.com. El servidor de nombre nos devolvería la lista de registros de tipo A, no uno en particular.

Así, en la implementación se itera por las respuestas, comprobando si alguna de ellas coincide con el valor indicado por el usuario, y dando error en caso contrario:

```

def check_dns(target, record_type, expected_address):

    """Checks if a certain domain has a DNS record (of the proper type)
    that matches the expected address"""

    # ...

    return_obj = {}

```

```

try:
    # Launch the query
    answer = dns.resolver.query(target, record_type)

    # The query finished properly
    return_obj['valid'] = True
    return_obj['status_ok'] = False

    # Check if any of the results matches the expected address
    for single_result in answer.rrset:
        ans = single_result.to_text().strip()

        if expected_address.strip() == ans:
            return_obj['status_ok'] = True

# Exception handling omitted
# ...

return return_obj

```

Implementación del chequeo de código de estado HTTP

Como se comentó en la sección 1.4.1, *Chequeo de peticiones HTTP*, las respuestas a las peticiones HTTP contienen un código de estado que da información sobre si la petición se procesó correctamente o no. Una de los tipos de código de estado son los códigos de redirección, aquellos que empiezan por el dígito 3xx.

Uno de los problemas que se encontraron tras la implementación inicial de este chequeo es que si un usuario usaba un código de esta familia en la comprobación, siempre daba error. Se descubrió que esto era así porque al hacer la petición, si el servidor devolvía un código de redirección, el sistema automáticamente seguía esa redirección y no devolvía el código al cliente. Por ello, se tuvo que añadir un caso especial para los códigos de la familia 3xx para que no se siguiesen las redirecciones.

```

def check_http_header(target, status_code=200):
    """Checks if a certain http URL returns the correct status code."""

    return_obj = {}

    try:
        # Don't follow redirections if status_code is in the 30x family
        if status_code / 10 == 30:
            r = requests.head(target, timeout=10)
        else:

```

```

r = requests.head(target, allow_redirects=True, timeout=10)

return_obj['valid'] = True
return_obj['status_code'] = r.status_code
return_obj['status_ok'] = r.status_code == status_code

# ...

```

6.3.2. Uso de herencia de modelos

Uno de los primeros dilemas que surgieron en el proyecto fue el usar o no herencia a la hora de definir los modelos de los chequeos. La motivación principal era que los chequeos, en general, tienen un gran número de parámetros y funciones comunes, por lo que resultaría interesante encontrar alguna forma de agrupar esos elementos comunes para evitar duplicidad de código.

Primera iteración: duplicación total

El primer enfoque que se intentó fue, directamente, repetir todos los campos y métodos en todos los chequeos. Dado que el proyecto cuenta con cuatro tipos de chequeos, los campos y métodos comunes se repetían cuatro veces, el código ocupaba mucho más espacio y era mucho más propenso a errores e inconsistencia.

Segunda iteración: herencia multitabla

El siguiente paso fue utilizar herencia directa (o *multitabla*) de modelos. Pongamos un ejemplo:

```

class BaseCheck(models.Model):
    title = models.CharField(max_length=50)
    target = models.CharField(max_length=50)

class PingCheck(BaseCheck):
    should_check_timeout = models.BooleanField()

```

En este caso, el modelo *PingCheck* es, a su vez, un tipo de *BaseCheck*. Además, es posible que existan *BaseChecks* de forma independiente. Esto es así porque Django crea dos tablas, una para cada modelo, y se genera una relación para los casos en los que un *BaseCheck* sea a la vez un *PingCheck*.

El problema en el caso de SiteUp es que no tendría sentido que existiese un chequeo base sin tipo, por lo que este enfoque no era válido. Además, la herencia multitabla está desaconsejada en la mayoría de los casos.

Tercera iteración: herencia abstracta

La herencia abstracta es similar a la herencia multitabla, pero con la diferencia de que para el *modelo base* no se genera una tabla, sino que los campos se guardan en la tabla de los modelos hijos. Modificando el caso anterior:

```
class BaseCheck(models.Model):
    title = models.CharField(max_length=50)
    target = models.CharField(max_length=50)

    class Meta:
        abstract = True

class PingCheck(BaseCheck):
    should_check_timeout = models.BooleanField()

class DnsCheck(BaseCheck):
    record_type = models.CharField(max_length=50)
```

En este caso, la tabla creada para el modelo *PingCheck* contendrá columnas para los campos *title*, *target* y *should_check_timeout*, y la tabla para el modelo *DnsCheck* contendrá los campos *title*, *target* y *record_type*. Además, cualquier método que se le añadiese al modelo base estaría disponible para los hijos.

Finalmente, esta fue el enfoque que se decidió usar. La única pega que aparece es que no hay manera sencilla de recuperar todos los objetos que hereden de un modelo – en el ejemplo, todos los objetos que heredan de *BaseCheck* – como sí la había en el caso de la herencia multitabla, pero es una pega menor, subsanable de forma sencilla utilizando métodos mágicos como `__subclasses__`:

```
lugares = sum((list(x.objects.all()) for x in BaseCheck.__subclasses__()), [])
```

6.3.3. Workflow de front-end

Para la implementación del front-end se han seguido las últimas prácticas en el desarrollo web front-end a nivel mundial. En particular, se ha utilizado un *tool-chain*¹ basado en Grunt [23], un sistema de ejecución de tareas desarrollado con node.js [33] pensado para automatizar tareas repetitivas o monótonas, como minificación, compilación, testeo, *linting*², etc.

En la figura 6.1 se puede ver una diagrama del flujo de desarrollo de las hojas de estilo y del código JavaScript de la parte frontend. El desarrollador trabaja dentro de la carpeta `siteup_frontend/static_src`, que contiene los ficheros fuente de las hojas de estilo y los scripts de JavaScript.

¹Toolchain: conjunto de herramientas que funcionan de forma encadenada y sucesiva.

²Linting: revisión de código para comprobar que sigue ciertas prácticas establecidas a nivel de lenguaje, organización, etc.

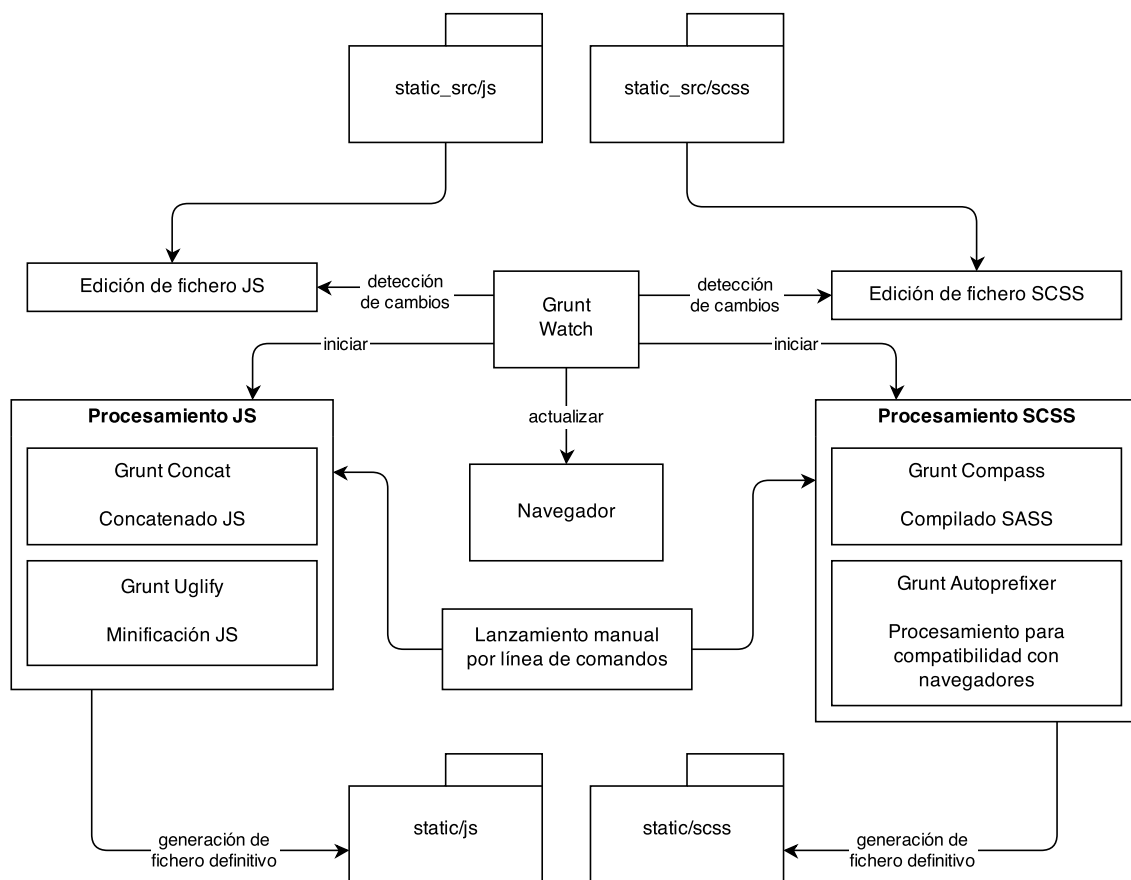


Figura 6.1: Diagrama de flujo de trabajo del front-end

Se detallan a continuación los pasos que se siguen en el flujo de trabajo.

Instalación de paquetes de desarrollo

Para seguir este flujo de trabajo es necesario instalar ciertas dependencias en el sistema. En particular, es necesario instalar el entorno node.js, según las instrucciones de su distribución particular. Hecho esto, lo siguiente es ejecutar el siguiente comando:

```
$ npm install
```

Esto generará un directorio `node_modules` con las dependencias de node instaladas.

Inicio del flujo de trabajo

El siguiente paso es lanzar uno de los plugins de grunt, **grunt-contrib-watch**, que se encarga de vigilar los cambios sobre los ficheros. Cuando detecta cambios, hace dos cosas:

- Primero, lanza las tareas de procesamiento que se detallan a continuación.
- Segundo, terminadas las tareas, lanza una señal para actualizar el navegador. Así se evita al usuario tener que ir al navegador y manualmente actualizar la web. Este proceso se conoce como **livereload**.

Este proceso se lanza utilizando el siguiente comando:

```
$ grunt watch  
Running "watch" task  
Waiting...
```

También es posible lanzar el procesamiento de forma manual ejecutando el siguiente comando:

```
$ grunt
```

El primer paso del procesamiento es siempre **limpiar** los ficheros generados previamente. Esta acción la lleva a cabo el plugin **grunt-contrib-clean**.

Procesamiento JS

El procesamiento de los ficheros JavaScript consta de varios pasos. El primero de ellos es la **concatenación** de todos los ficheros de código. Es una buena práctica tener el menor número de ficheros anexos en una web para reducir al mínimo el número de peticiones al servidor. Para esto se utiliza el plugin **grunt-contrib-concat**, que navega por el directorio `siteup_frontend/static_src/js` y concatena todos los ficheros JavaScript que encuentre en un solo fichero.

El siguiente paso es la **minificación** del fichero generado. Básicamente lo que se busca es reducir el tamaño final del fichero mediante varios procesos, como la eliminación de espacios en blanco y la reducción de los nombres de las variables. Esto lo lleva a cabo el plugin **grunt-contrib-uglify**

Hecho esto, se copia el fichero generado a la carpeta de salida `siteup_frontend/static/js`. Una muestra de la salida generada por Grunt al realizar el procesamiento es la siguiente:

```
>> File "siteup_frontend/static_src/js/main.js" changed.

Running "clean:js" (clean) task
Cleaning siteup_frontend/static/js/main.js...OK

Running "concat:build" (concat) task
File "siteup_frontend/static/js/main.js" created.

Done, without errors.
Completed in 0.335s at Sat Apr 19 2014 19:00:22 GMT+0200 (CEST) - Waiting...
```

Procesamiento SCSS

Como se mencionó previamente en la sección 5.1.1, para el desarrollo de las hojas de estilo se utilizó Sass, un superconjunto de CSS que añade numerosas funcionalidades que habitualmente resultan tediosas de hacer directamente en CSS. El único contra que tiene el uso de Sass es que los ficheros no pueden ser interpretados directamente por los navegadores, sino que es necesario compilarlos a CSS tradicional.

Así, el primer paso del procesamiento es la compilación de los ficheros Sass. Esto lo lleva a cabo el plugin **grunt-contrib-compass**, que lee los ficheros situados en `siteup_frontend/static_src/scss` y realiza la generación.

El siguiente paso es el procesamiento del fichero CSS generado para maximizar su compatibilidad con todos los navegadores. Esto se hace leyendo las propiedades CSS utilizadas y añadiendo prefijos en aquellos casos que sea necesario. El plugin encargado de hacer este trabajo es **grunt-autoprefixer**. Como ejemplo, si tenemos el siguiente código original:

```
a {
  transition: transform 1s
}
```

Tras el procesamiento se convertirá a:

```
a {
  -webkit-transition: -webkit-transform 1s;
  transition: -ms-transform 1s;
```



```

    transition: transform 1s
}

```

Se ve que el plugin añade los prefijos necesarios para los navegadores basados en WebKit y para Internet Explorer (-webkit y -ms respectivamente).

Resulta interesante comentar que este plugin es configurable, de forma que podamos decidir a qué navegadores vamos a dar soporte. Por defecto añade los prefijos necesarios para soportar las dos últimas versiones de los navegadores más populares.

Tras este proceso, el fichero generado se copia a la carpeta `siteup_frontend/static/css`.

6.4. Gestión de la base de datos

El uso del mapeo objeto-relacional (ORM) que integra Django ayuda a que el desarrollador no tenga que interactuar con la base de datos utilizando lenguajes de bajo nivel como SQL (Structured Query Language), sino que se trabaja con una capa de abstracción mayor. Es más, la creación de las estructuras de la base de datos (tablas, índices, restricciones de integridad, etc.) también la lleva a cabo el ORM, en particular a través de scripts de sincronización y migraciones.

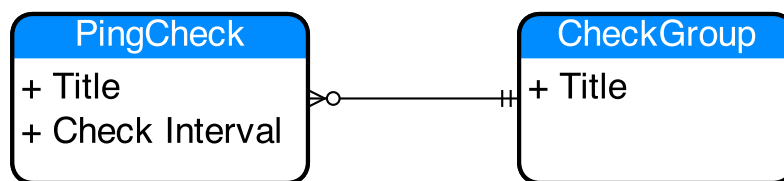


Figura 6.2: Diagrama ER de ejemplo 1

Pongamos por ejemplo un sencillo esquema entidad-relación como el que aparece en la figura 6.2, en el que se definen dos entidades con una relación de tipo 1:N. Haciendo uso del ORM de Django, este esquema se implementaría de la siguiente manera:

```

from django.db import models

class CheckGroup(models.Model):
    title = models.CharField(max_length=100)

class PingCheck(models.Model):
    title = models.CharField(max_length=100)
    check_interval = models.PositiveSmallIntegerField()

    group = models.ForeignKey(CheckGroup)

```

Una vez definidos los modelos, la base de datos se generará utilizando el siguiente comando de Django:

```
$ python manage.py syncdb
```

El comando syncdb *sincroniza* la base de datos con los modelos, creando las tablas necesarias. Desde este momento, el ORM de Django ofrecerá una serie de comandos para hacer consultas de manera sencilla. Por ejemplo, para obtener todos los grupos de chequeos:

```
grupos = CheckGroup.objects.all()
```

Para obtener el grupo llamado *Personal*:

```
grupo = CheckGroup.objects.get(title="Personal")
```

Es posible embeber condiciones algo más avanzadas. Por ejemplo, para obtener los chequeos que se ejecutan cada 3 minutos o menos:

```
chequeos_frecuentes = PingCheck.objects.filter(check_interval__lte=3)
```

Django detecta las relaciones entre modelos y provee mecanismos para navegar esas relaciones, por ejemplo.

```
grupo = CheckGroup.objects.get(title="Personal")
chequeos_personales = grupo.pingcheck_set.all()
```

```
chequeo = PingCheck.objects.get(title="Web personal")
grupo = PingCheck.group
```

Django automáticamente detecta las relaciones y genera las consultas necesarias para unir los datos de las tablas involucradas. Django también cumple con las restricciones de integridad que se impongan en la base de datos. En nuestro ejemplo, si borramos un grupo, todos los chequeos del grupo son borrados también de forma automática.

```
>>> grupo = CheckGroup.objects.get(title="Personal")
>>> grupo.delete()
>>> chequeo = PingCheck.objects.get(title="Web personal")
Traceback (most recent call last):
```

```
...
DoesNotExist: PingCheck matching query does not exist.
```

Este comportamiento puede configurarse para que no se produzca el borrado *en cascada*, al igual que muchísimos otros parámetros que también son configurables.

Django también ofrece métodos para comprobar que los datos de un modelo son correctos antes de guardarlo en la base de datos. Por ejemplo, si intentamos introducir una dirección de correo inválida para un usuario, el sistema nos alertará:

```
>>> usuario = User()
>>> usuario.username = "Fab"
>>> usuario.email = "invalido"
>>> usuario.full_clean()
Traceback (most recent call last):
```

```
...
ValidationError: {'email': [u'Enter a valid email address.']}
```

6.4.1. Cambios en la estructura

El problema del proceso planteado ahora es que tiene una limitación: no contempla cambios estructurales en los modelos una vez que la base de datos esté creada. Para ello es necesario utilizar **South**, tal y como se comentó en la sección 6.1.1.

South compara el estado de la base de datos con los nuevos modelos, detecta los cambios y realiza las operaciones necesarias. Si, por ejemplo, añadiésemos un atributo *description* al modelo *CheckGroup*, éste quedaría así:

```
class CheckGroup(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
```

Ahora, para evitar perder los datos que ya tuviésemos en la base, utilizaríamos South para crear una **migración**:

```
$ python manage.py schemamigration aplicacion --auto
+ Added field description on main.CheckGroup
Created 0002_auto__add_field_group_description.py.
```

```
$ python manage.py migrate aplicacion
Running migrations for main:
- Migrating forwards to 0002_auto__add_field_group_description.
> main:0002_auto__add_field_group_description
- Loading initial data for main.
Installed 0 object(s) from 0 fixture(s)
```

En caso de que borrásemos el campo o quisiésemos hacer más cambios, estas migraciones pueden deshacerse o modificarse sin mayor problema.

Capítulo 7

Pruebas

En este capítulo se detallan las baterías de pruebas a las que se ha sometido **SiteUp**, ya sean de carácter manual o automatizadas mediante software específico.

7.1. Estrategia

La estrategia de pruebas seguida en SiteUp es híbrida. Por un lado, se cuenta con una batería de pruebas automatizada, basadas en el motor de pruebas que incluye el framework de desarrollo, que verifican las funcionalidades más críticas del sistema.

Por otro lado, se cuenta con pruebas manuales realizadas de forma periódica. A corto plazo y de forma frecuente se revisaban las funcionalidades más inmediatas de las aplicaciones. A largo plazo, se han dado de alta numerosos chequeos y se han verificado a lo largo del tiempo.

7.2. Entorno de pruebas

Para las pruebas automáticas, el entorno de pruebas es una copia virtual del entorno de producción real, creado automáticamente por el software de testeo, de forma que no se modifiquen los valiosos datos de la base de datos de producción. En cada ejecución de los tests se crea una base de datos y un entorno nuevos, vacíos, en los que se llevan a cabo las pruebas.

Para las pruebas manuales, dado que se realizan directamente sobre el sistema en producción, el entorno coincide con lo dispuesto en la sección 5.1.1.

7.3. Roles

Se presentan dos roles principales necesarios para la ejecución de las pruebas.

7.3.1. Desarrollador principal

En primer lugar, el desarrollador del producto es el principal probador del software. Por un lado, es el encargado del desarrollo y ejecución de las baterías de pruebas automatizadas, teniendo la obligación de interpretar los resultados y actuando en consecuencia.

Por otro lado, también ha de probar el producto manualmente como un usuario más, sobre todo a tenor de lo expuesto en la sección 1.2, en la que se refleja que la motivación principal del proyecto es una necesidad personal del desarrollador.

7.3.2. Probadores externos

Además del desarrollador principal fue conveniente contar con la ayuda de varios probadores externos que dieran su punto de vista sobre el software, como usuarios. Así, realizaron pruebas de carácter manual a corto plazo, opinando sobre la interfaz de usuario, la usabilidad y la accesibilidad del proyecto, así como a largo plazo, dando de alta diversos chequeos y probando su funcionalidad.

En todas las etapas, el desarrollador obtenía feedback de los probadores, integrando en la medida de lo posible los consejos y apreciaciones que se obtenían.

7.4. Pruebas funcionales

Las pruebas funcionales engloban las pruebas que tratan de comprobar el correcto cumplimiento de los requisitos funcionales, de forma directa (verificando su cumplimiento de forma directa) como de forma indirecta a varios niveles (verificando que las dependencias de las que pende la funcionalidad funcionen correctamente).

En primer lugar, se usaron **pruebas unitarias**, que tuvieron por objetivo localizar errores en los elementos software antes de ser integrados con el resto de elementos del sistema.

En SiteUp, el grueso de las pruebas unitarias se centró principalmente en el módulo de chequeo, organizado dentro de la aplicación `siteup_checker`. Los diversos casos de test comprobaron de forma individual que los procedimientos que llevan a cabo los chequeos devuelvan el resultado correcto, asegurando la ausencia de falsos positivos y resultados erróneos.

Por ejemplo, para la verificación del procedimiento de **chequeo HTTP** se hizo uso del sitio web **HttpBin** [26], una web especialmente pensada para facilitar el testeo de conexiones HTTP.

La implementación de las pruebas se ha hecho utilizando las capacidades de testing que provee el framework Django. Las pruebas unitarias se lanzan a través de la línea de comandos, indicando el módulo de test unitario que se quiere comprobar, por ejemplo:

```
$ python manage.py test siteup_checker.tests.test_http
Creating test database for alias 'default'...
```

(output omitted)

```
-----
Ran 7 tests in 2.506s
```

OK

```
Destroying test database for alias 'default'...
```

Tras la ejecución de los tests se muestra una lista de aquellos que han fallado, ya sea por no haberse cumplido los asertos definidos o por otro error no controlado.

Por otro lado, Las **pruebas de integración** tuvieron por objetivo localizar errores en módulos completos, analizando la interacción entre varios artefactos software.

Como ejemplo, tras realizar las pruebas unitarias en los módulos de chequeo, se pasó a hacer pruebas de integración entre los procedimientos de chequeo y las instancias de los modelos de chequeo dadas de alta por los usuarios, de forma que se asegurase que la creación de un chequeo por parte de un usuario era reflejada en la base de datos y daba lugar a la correcta ejecución del procedimiento de chequeo. La forma de implementar estas pruebas fue, de nuevo, utilizando el framework de testeo de Django.

7.4.1. Pruebas de sistema

Las pruebas de sistema, que buscan asegurar que el sistema cumple con todos los requisitos establecidos, se desarrollaron de forma continua a medida que iba avanzando el proyecto.

En cada iteración, las pruebas se llevaron a cabo utilizando un entorno virtual y el servidor de pruebas que integra el propio framework Django, y se utilizó software de *throttling* de red para simular condiciones de red adversas que se asemejasen a las condiciones reales.

7.4.2. Pruebas de aceptación

Para verificar que el producto estaba listo para el paso a producción se hizo un despliegue en un servidor real, ubicado en Alemania, y se dio de alta una decena de chequeos de todos los tipos. El servidor ha permanecido online desde mediados del mes de febrero hasta la fecha.

En este tiempo, además de recibir las actualizaciones del software a medida que el desarrollo avanzaba, el proyecto ha detectado de forma eficiente las caídas en el servicio de los chequeos dados de alta, notificando de manera correcta tanto por correo electrónico como a través de la aplicación de Android.

En particular se puede poner de ejemplo la web oficial de la Universidad de Cádiz, que se ha utilizado como objetivo de los chequeos. En el tiempo que ha estado el chequeo activo se han detectado caídas en la web unas 15 veces, la última el 16 de abril de 2014, día en el que durante la mañana la web ha pasado a estar inaccesible más de 7 veces durante periodos de entre 10 y 30 minutos.

7.5. Pruebas no funcionales

Las pruebas no funcionales buscan verificar aspectos del proyecto más allá de su funcionalidad, tales como la seguridad del sitio, su estabilidad, la accesibilidad, etcétera.

7.5.1. Pruebas de seguridad

Las pruebas de seguridad se centraron en verificar que la información de los usuarios y las áreas privadas del sitio web son seguras. Entre las pruebas que se realizaron se encuentran las siguientes:

- Se intentó acceder a URLs de la plataforma web sin haber iniciado sesión, para comprobar que el control de usuarios es correcto. Por ejemplo, intentando acceder sin haber hecho login a la URL

`http://siteup.josetomastocino.com/dashboard`

La plataforma web actuó correctamente en estos casos, redirigiendo al usuario a la pantalla de login.

- Se intentó obtener información de chequeos de otros usuarios, para ver si había filtración de datos, cambiando manualmente los identificadores enviados. Por ejemplo, la siguiente URL

`http://siteup.josetomastocino.com/checks/pingcheck/6/`

pertenece a la página de detalle del PingCheck con identificador 6, perteneciente a un usuario. Si intentamos acceder a esa URL con otro usuario,

la plataforma web detecta este intento y no muestra el recurso, sino que muestra un error 404.

- Se intentaron llevar a cabo varias inyecciones sql [53], consistentes en escribir código SQL malicioso en los formularios de la plataforma web para ver si llegaban a ejecutarse en la base de datos. La plataforma web a través de los sistemas de saneado de formularios con los que cuenta consiguió que estos intentos no tuvieran efecto.

7.5.2. Pruebas de fiabilidad

Se comprobó satisfactoriamente el cumplimiento del requisito no funcional NRQ-3, que impone que los retrasos en el lanzamiento de chequeos no pueden superar el 200 % del tiempo entre chequeos.

Para ello, se añadió una carga importante de chequeos en el sistema y se monitorizaron las operaciones de Celery. En todos los casos la cola de prioridad del sistema de tareas asíncronas permitió que los chequeos se ejecutaran dentro de su *slot* de tiempo adecuado, nunca superando la limitación previamente dispuesta.

7.5.3. Pruebas de accesibilidad

El requisito no funcional NRQ-4 dictaminaba que *la plataforma web deberá ser accesible desde cualquier clase de dispositivo móvil con acceso a internet*. Para verificar su cumplimiento se probó la plataforma web en los siguientes dispositivos, siendo satisfactoria la navegación en todos ellos:

- Ordenador de mesa con navegadores Google Chrome, Firefox y IE 11, y resoluciones hasta 1024x768.
- Tablet Google Nexus 7 de primera generación.
- Samsung Galaxy Nexus.
- Jiayu G2.
- Apple iPad 2

Se presentan en las figuras 7.1, 7.2 y 7.3 algunas capturas de pantalla de la web en diversos dispositivos.

7.5.4. Verificación de la calidad del código

En la medida de lo posible se ha seguido la guía de estilo oficial de Python, expuesta en el documento oficial PEP8 [35] por el propio creador del lenguaje, Guido van Rossum. Para reforzar el seguimiento de estas buenas prácticas de código se ha hecho uso de herramientas de verificación automática, que comprueban que el

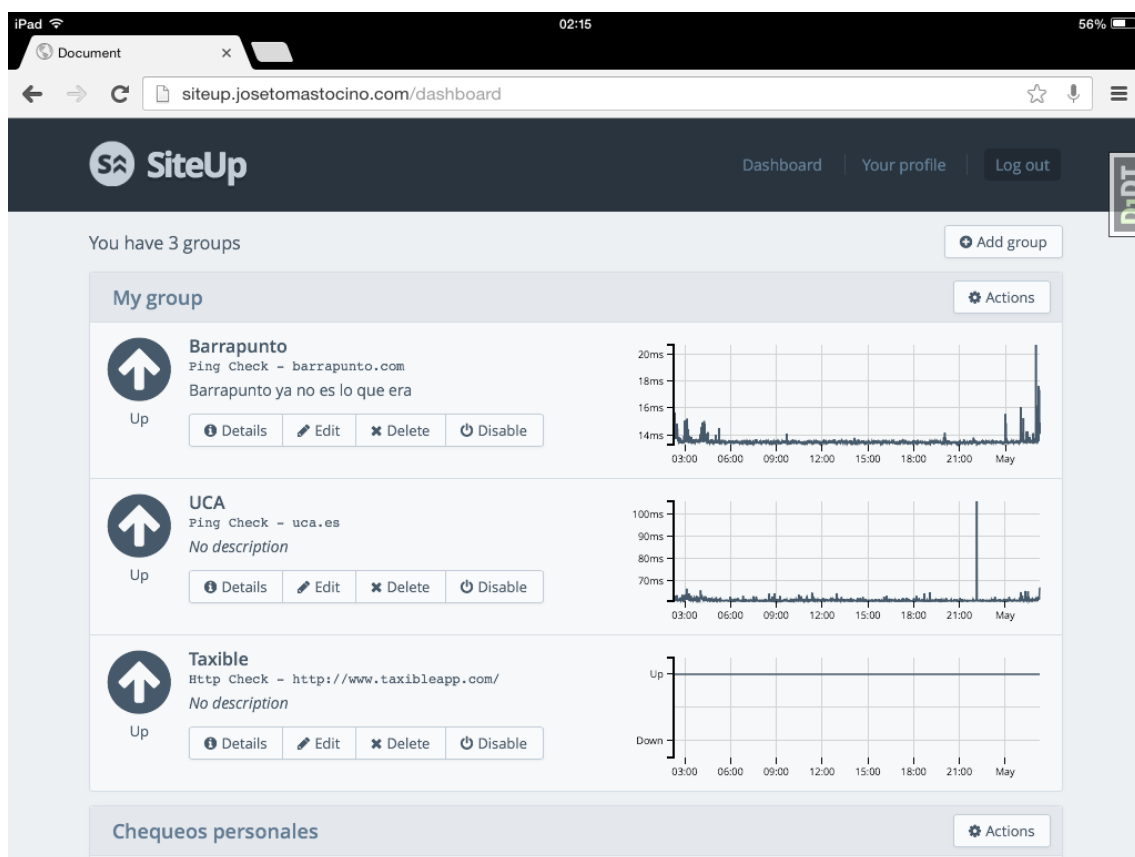


Figura 7.1: Plataforma web vista mediante un Apple iPad 2

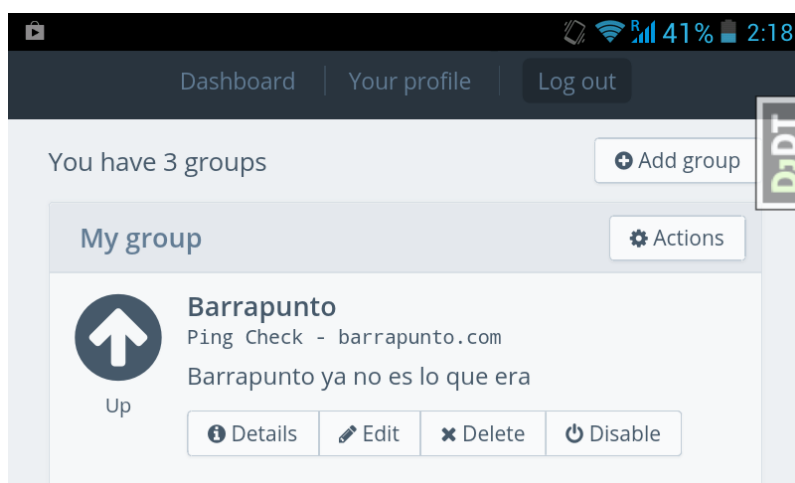


Figura 7.2: Plataforma web vista mediante un Jiayu G2 (orientación horizontal)

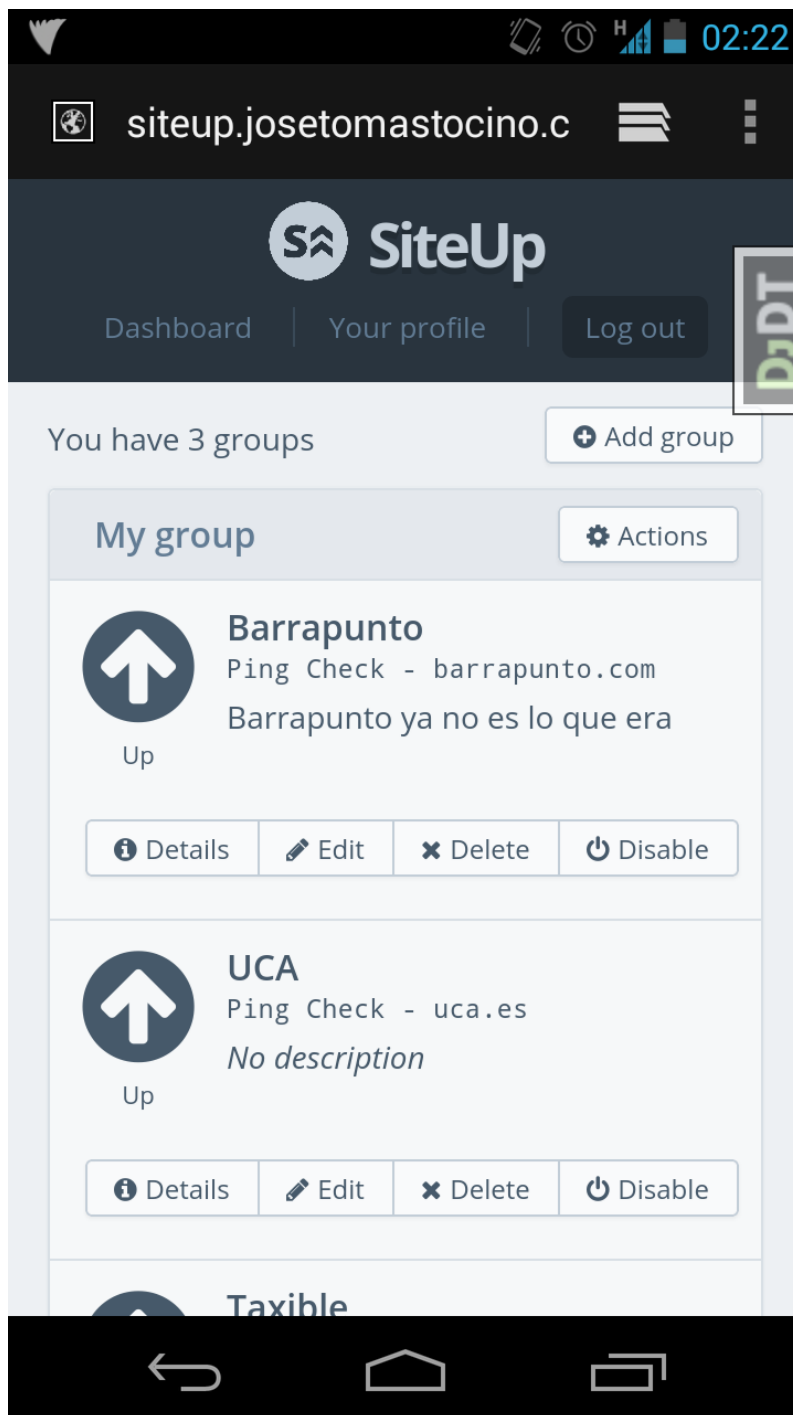


Figura 7.3: Plataforma web vista mediante un Galaxy Nexus

código se ajuste a las convenciones de nombrado, sintaxis y otras recomendaciones presentes en el citado documento.

Capítulo 8

Conclusiones

Durante el transcurso del desarrollo de SiteUp, y sobre todo al término del mismo, se han obtenido unas conclusiones y unos resultados, tanto de forma personal como para con la comunidad, que intentaremos reflejar en este capítulo.

8.1. Objetivos cumplidos

Al término del desarrollo del proyecto, el proyecto ha completado todos los objetivos a cumplir, presentes en el planteamiento inicial y detallados en la sección 1.3. En particular:

- Se ha creado un conjunto de herramientas de chequeo de servicios de Internet, que se encuentra disponible en el módulo `siteup_checker/monitoring` del proyecto.
- Se ha creado una aplicación online de acceso público con la que los usuarios pueden gestionar sus chequeos de manera sencilla, que en la fecha de escritura de la presente memoria es accesible en la url <http://siteup.josetomastocino.com>.
- Se ha establecido un sistema de notificaciones con el que mantener informados a los usuarios tanto por correo electrónico como mediante una aplicación móvil para el sistema operativo Android desarrollada a tal efecto. Actualmente la aplicación se encuentra disponible en la forja de código [27] y en un futuro lo estará en el Google Play Store.

La compleción total de los objetivos funcionales también pone de manifiesto que se han alcanzado satisfactoriamente los objetivos transversales y personales dispuestos en el inicio. Se han afianzado fuertemente los conocimientos de desarrollo web en general y con Django en particular hasta un nivel de competencia suficiente para servir como baza en la búsqueda laboral. Además, SiteUp ha servido satisfactoriamente como primera incursión en el mundo del desarrollo de aplicaciones Android.

8.1.1. Limitaciones del proyecto

Aunque cubre una gran parte de los puntos de vigilancia habituales, la aplicación se limita a ofrecer chequeo de respuesta de ping, chequeo de puertos, chequeo de registros DNS y chequeo de cabeceras y contenidos HTTP.

La aplicación de Android no cuenta con ninguna funcionalidad para gestionar los chequeos de un usuario, sino que sirve para recibir notificaciones instantáneas provenientes de la aplicación web. Ésta, por otro lado, está completamente adaptada para su uso a través de dispositivos móviles gracias al uso del *responsive web design*.

Idealmente los chequeos deberían hacerse simultáneamente desde diferentes máquinas colocadas en diversos puntos geográficos, para así tener unos resultados más fiables. La falta de infraestructuras y la finalidad didáctica del proyecto limitan la aplicación a una estructura monolítica en la que los chequeos se hacen desde una sola máquina, la misma que sirve el servicio web.

8.2. Conclusiones personales

SiteUp es uno de los pocos proyectos personales que han surgido para suplir una **necesidad** real y específica y que, al término del desarrollo, han conseguido su objetivo. La idea inicial, expuesta en la sección 1.2, ha servido como eje motor del proyecto en todo momento. Si se diese la situación presentada en el inicio, el proyecto SiteUp sería capaz de detectar el problema e informar en consecuencia.

8.2.1. Lecciones aprendidas

La cantidad de tecnologías diferentes utilizadas en cada uno de los niveles de abstracción del proyecto es bastante grande. Adquirir una competencia básica en todas estas tecnologías es una tarea de una envergadura importante, y la correcta ejecución del sistema es comparable al mecanismo de un reloj, en el que todas las piezas deben funcionar de manera coordinada.

Desde la configuración de los numerosos servicios usados en el entorno de producción, al despliegue de la aplicación, pasando por el desarrollo de los módulos de chequeo, hasta llegar a la implementación del código de la capa de presentación, sin olvidar la implementación de la aplicación Android. En todos los niveles, en unos en mayor medida que en otros, se ha alcanzado una soltura suficiente para el desarrollo y posterior lanzamiento de un producto completo.

Al tratarse de un proyecto de ejecución continua en el que constantemente se están haciendo chequeos y enviando notificaciones, el flujo de *feedback* entre los usuarios y el desarrollador (que en las etapas iniciales son la misma persona) es mucho más **dinámico** que en cualquier otro proyecto en el que la ejecución fuese

momentánea, como un juego o una aplicación de escritorio. Este planteamiento ha permitido que en SiteUp se hayan detectado rápidamente conflictos y problemas en el sitio, como por ejemplo que algunos usuarios prefieren no recibir resúmenes diarios de sus chequeos, motivando la inclusión de una opción en sus perfiles para desactivar esta funcionalidad.

Finalmente, tras la conclusión del proyecto, se ha conseguido un **punto de vista** diferente de las alternativas presentadas en la sección 1.4, *Estado del arte*. Con un conocimiento más formado de lo que hay detrás (salvando las distancias) de estos proyectos resulta más sencillo **evaluar** en qué aciertan y en qué fallan, qué les falta y qué les sobra, y sobre todo qué se puede aprender de ellos para su integración en SiteUp. En particular, me ha sorprendido que no sea común que esta clase de servicios proporcionen una aplicación móvil nativa que sirva, al menos, para la recepción de notificaciones, tal y como se hace en SiteUp. La inversión es despreciable si la comparamos con la utilidad que provee.

8.3. Difusión y reconocimiento

Gracias a SiteUp tuve la oportunidad de participar en el *VIII Concurso Universitario de Software Libre* [43], pudiendo presentar el proyecto a la comunidad universitaria local el miércoles 23 de abril de 2014 en la Universidad de Cádiz. En la presentación obtuve opiniones y consejos para mejorar en el proyecto que en mayor o menor medida se han acabado tomando en cuenta e integrando.

El jurado del premio local decidió nombrar a SiteUp ganador del premio al *Mejor Proyecto Innovador de Software Libre* [30].

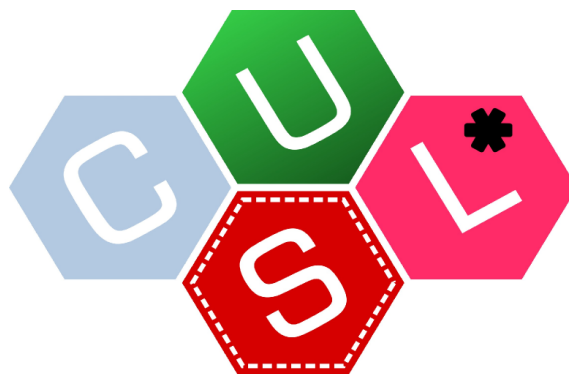


Figura 8.1: Logotipo VIII CUSL

8.4. Trabajo futuro

Durante la etapa de desarrollo y, sobre todo, durante las sesiones de pruebas han surgido muchos frentes de trabajo futuro, unos de mayor relevancia que otros, pero importantes al fin y al cabo y que aquí se presentan.

8.4.1. SiteUp Probes

SiteUp Probes es un proyecto que busca crear *sondas* que ejecuten de manera independiente los chequeos dados de alta en una instalación de SiteUp, enviando los resultados de vuelta al servidor central. La idea es que estas sondas esté geográficamente distribuidas, de forma que los resultados de los chequeos no dependan tanto de la situación del nodo central, sino que se calculen como una media de los resultados obtenidos por distintas sondas.

La motivación de esta ampliación viene dada por lo descrito en la sección 2.6.1, *Compromiso de la plataforma de despliegue*. El uso de un solo punto a la hora de lanzar los chequeos puede comprometer la veracidad y corrección de los resultados.

Se trata de un escalón importante y complejo en el trabajo de ampliación del proyecto. A nivel de implementación implicaría habilitar alguna clase de canal de comunicación entre el nodo central de SiteUp y las sondas para la comunicación de los chequeos a ejecutar y de los resultados de éstos. A nivel logístico conllevaría la búsqueda de nuevos sistemas en los que instalar las sondas, más allá del servidor de producción que está alquilado para el proyecto.

8.4.2. Monetización

Para asegurar la supervivencia de cualquier proyecto es necesario contar alguna clase de **modelo de negocio** que sufrague los gastos que se generan [47]. En el caso de SiteUp, la **monetización** se puede conseguir mediante la creación de niveles de cuentas de usuarios, que gocen de distintos beneficios. Por ejemplo:

- Las cuentas gratuitas solo podrán crear un chequeo, y el tiempo entre chequeos será como mínimo de 5 minutos.
- Las cuentas básicas costarán 5€ al mes, podrán crear hasta 5 chequeos y el tiempo mínimo entre chequeos será de 2 minutos.
- Las cuentas premium costarán 10€ al mes, podrán crear hasta 30 chequeos y el tiempo mínimo entre chequeos será de 1 minuto.

La implementación de este modelo de monetización no es compleja, aunque la gestión de pasarelas de pago y cobros recurrentes es una de los procedimientos más problemáticos a nivel de programación, sobre todo a la hora de verificar la correcta ejecución del sistema.

8.4.3. Ampliación de los tipos de chequeos

Los cuatro tipos de chequeos actualmente presentes en SiteUp cubren un gran abanico de frentes a la hora de vigilar el estado de un servicio de Internet, pero

existen (y seguirán surgiendo) nuevos puntos que convendría comprobar. Como ejemplos de posibles chequeos a añadir en un futuro se podrían considerar:

- Chequeo de servidores de bases de datos mediante el envío de consultas de verificación.
- Chequeo de transacciones. Esto es, chequeos compuestos de varios pasos.
- Chequeos de APIs con campos adicionales, como autenticación, cabeceras especiales o parámetros GET/POST.

El trabajo en este punto pasaría por codificar los nuevos modelos de forma similar a los que ya existen, que heredan de un modelo base que hace sencilla la extensión del sistema.

8.4.4. Ampliación de tipos de notificación

Actualmente, SiteUp envía notificaciones mediante correo electrónico y mediante la aplicación Android. Como posibles ampliaciones se podrían considerar el uso de mensajes cortos (SMS) y el desarrollo de una app para iOS capaz de recibir notificaciones PUSH.

Apéndice A

Manual de instalación de la plataforma web

A continuación se presentan las instrucciones de instalación de la plataforma web en un servidor de producción. La versión más actualizada de estas instrucciones se pueden encontrar en el fichero `INSTALLING-web.md` alojado en el directorio raíz de la forja de código [27].

Antes de continuar se presentan los requisitos de hardware y software mínimos para la correcta ejecución de la plataforma web.

Hardware

- 512MiB de memoria RAM como mínimo.
- 10GiB de disco duro como mínimo.
- Acceso a Internet con un canal de subida de al menos 1Mbit/s.

Software

- Sistema operativo **GNU/Linux**, preferiblemente basado en paquetería Debian.
- Servidor de shell remota **SSH**, accesible desde el exterior.
- **Nginx**, servidor web trabajando en modo de proxy inverso.
- **Supervisord**, sistema para el control de procesos que trabajen en modo demonio..
- **RabbitMQ**, cola de tareas sencilla que cumple el estandar AMQP.
- Intérprete de **Python**, versión mínima 2.7.
- Soporte de entornos virtuales **VirtualEnv** para la encapsulación de dependencias.

A.1. Instalación inicial

A.1.1. Descarga de código

Para instalar la plataforma web es necesario clonar el repositorio [27] con el código de la aplicación. Primero, creamos una ubicación donde alojar el código desde la terminal:

```
$ mkdir /srv/siteup -p
$ cd /srv/siteup
```

Una vez ahí, instalamos el software de control de versiones **git** y clonamos el repositorio desde Github:

```
$ sudo apt-get install git
$ git clone https://github.com/JoseTomasTocino/pfc-ii.git .
```

A.1.2. Entorno virtual y dependencias

Como se ha comentado previamente, el proyecto utiliza **virtualenv** y **virtualenvwrapper** para una gestión más limpia de las dependencias. La instalación de estos dos elementos es sencilla, en primer lugar se instala el gestor de paquetes y los dos paquetes mencionados:

```
$ sudo apt-get install python-pip
$ sudo pip install virtualenv
$ sudo pip install virtualenvwrapper
```

En segundo lugar, añadimos el código de *bootstrapping* de **virtualenvwrapper** a nuestro perfil de terminal, habitualmente `.bashrc`:

```
$ cat >> ~/.bashrc

if [ -f /usr/local/bin/virtualenvwrapper.sh ]
then
    source /usr/local/bin/virtualenvwrapper.sh
fi

EOF
```

Hecho esto, será necesario reiniciar la terminal, tras lo cual podremos crear el entorno virtual e instalar las dependencias:

```
$ mkvirtualenv siteup
$ sudo apt-get install python-dev
$ pip install -r web/requirements.txt
```

El proceso de instalación de dependencias puede tardar entre 5 y 10 minutos aproximadamente.

A.1.3. Instalación de credenciales

Antes de continuar es necesario dar valor a algunas variables de credenciales privadas que el proyecto necesita. En particular, se deben definir dos variables:

- `EMAIL_PASS`, es la clave de la cuenta de correo utilizada para enviar notificaciones. Se define en el fichero `web/siteup/settings/base.py`.
- `GCM_API_KEY`, es la clave de la API del servicio Google Cloud Messaging.

Para el proceso de instalación y en tiempo de desarrollo lo habitual es añadirlas **script de activación** del entorno virtual, de la siguiente manera:

```
$ cdvirtualenv
$ cat >> bin/postactivate
export EMAIL_PASS=yourpass
export GCM_API_KEY=yourkey
```

EOF

Para la puesta en producción, esas variables se configurarán un poco más adelante.

A.1.4. Creación de la base de datos

El siguiente paso es preparar la base de datos. Volvemos al directorio en el que estábamos antes y ejecutamos el siguiente comando, que genera la estructura básica de la base de datos para las tablas y aplica las migraciones existentes:

```
$ cd /srv/siteup
$ python web/manage.py syncdb
$ python web/manage.py migrate
```

Tras esto, la base de datos se habrá generado y se encontrará en el fichero `web/db/django-db.sqlite3`.

A.1.5. Disposición de ficheros estáticos

Es importante que los ficheros estáticos de todos los componentes de la aplicación se encuentren en un mismo lugar. Para ello, Django provee un comando que unifica todos los estáticos en la carpeta indicada.

```
$ python web/manage.py collectstatic --noinput
```

A.1.6. Instalación del servidor

En estas instrucciones vamos a usar **Nginx** como servidor frontal, aunque también es posible utilizar Apache. Para instalar nginx, en sistemas GNU/Linux basados en

paquetería Debian, el procedimiento es muy sencillo:

```
$ sudo apt-get install nginx -y
```

A.1.7. Instalación del bróker de mensajes

El bróker de mensajes elegido es **RabbitMQ**, aunque en principio valdría cualquier otro bróker que fuese compatible con el estándar AMQP. La instalación es sencilla:

```
$ sudo apt-get install rabbitmq-server -y
```

La configuración por defecto es suficiente para las necesidades del proyecto.

A.1.8. Ejecución de los servicios

Para el control de los servicios que ejecutan el proyecto es necesario utilizar alguna clase de software de monitorización de procesos. La opción recomendada es **supervisor**. Su instalación es sencilla:

```
$ sudo apt-get install supervisor -y
```

Tras esto, es necesario generar el fichero de configuración de supervisor. SiteUp integra una tarea que automáticamente genera este fichero por nosotros. Solo tendremos que indicar las dos variables de credenciales que se han comentado previamente, simplemente introduciendo su valor por teclado cuando nos sea preguntado.

Para generar el fichero de configuración utilizaremos el siguiente comando:

```
$ python web/manage.py supervisor_conf
```

Se generará un fichero `supervisor-siteup.conf` con la configuración. Es buena idea revisar el fichero para comprobar que todas las rutas son correctas. Tras ello, solo queda mover el fichero al lugar correcto y reiniciar el demonio de supervisor.

```
$ sudo mv web/supervisor-siteup.conf /etc/supervisor/conf.d
```

A.1.9. Configuración del servidor frontal

El siguiente paso es configurar el servidor nginx para que reciba las peticiones apropiadas. Vamos a suponer que SiteUp va a recibir las peticiones en el dominio `siteup.uca.es`. Creamos el fichero `/etc/nginx/sites-available/siteup` y añadimos el siguiente contenido:

```
server {  
    listen 80;  
    server_name siteup.uca.es;
```

```
# Static files
location /static/ {
    root /srv/siteup/web;
    try_files $uri $uri/ =404;
}

location / {
    proxy_set_header X-Real-IP      $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host          $http_host;
    proxy_redirect off;

    if (!-f $request_filename) {
        proxy_pass      http://127.0.0.1:8000;
        break;
    }
}
}
```

Tras eso, activamos el fichero de configuración de la siguiente manera:

```
$ cd /etc/nginx/sites-enabled
$ sudo ln -s ../sites-available/siteup
```

Básicamente lo que hace ese fichero de configuración es revisar las peticiones que llegan y, si apuntan a la ruta /static, nginx se encarga de servir esos ficheros estáticos de front-end. Si no, pasa la petición al servidor gunicorn que estará escuchando en la dirección local 127.0.0.1:8000.

A.1.10. Lanzamiento final

Tras toda esta configuración, los últimos pasos que quedan son activar los servicios y servidores. Primero, lanzamos el controlador supervisor, que a su vez lanzará el servidor dinámico gunicorn y el servidor de tareas celery:

```
$ sudo service supervisor restart
```

Seguidamente, reiniciamos el servidor frontal para que pueda empezar a recibir peticiones:

```
$ sudo service nginx restart
```

Con esto, el proyecto estará funcionando y debería ser accesible a través del navegador.

A.2. Pruebas de implantación

Para probar que la instalación es correcta, lo ideal es que pruebe la aplicación a fondo, revisando toda la funcionalidad. El itinerario que se recomienda es el siguiente. Si tiene dudas sobre cómo realizar alguna de las acciones descritas, revise el Manual de usuario.

El primer paso antes de nada es lanzar la batería de tests que viene incluida en el proyecto, mediante el comando

```
$ python web/manage.py test
```

Todos los tests deberían funcionar correctamente. Si no lo hacen, es que ha habido un fallo en el sistema o en la aplicación.

Una vez descartados esos problemas, pruebe que el **registro y acceso de usuarios** funciona creando un nuevo usuario y accediendo a su cuenta. Si este procedimiento funciona, se habrá verificado que la instalación básica de la base de datos es correcta.

El siguiente paso es **crear un grupo de chequeos** y posteriormente crear un chequeo de tipo Ping básico. Se recomienda utilizar alguna web conocida como objetivo, por ejemplo la web de la Universidad de Cádiz en su dirección `uca.es`.

Tras crear el chequeo, éste aparecerá en modo **Fetching**, durante el que se están obteniendo los datos iniciales. Espere dos o tres minutos y vuelva a revisar su estado. Si sigue en modo *Fetching*, probablemente las tareas de chequeo no se estén llevando a cabo, lo que seguramente sea causado por una mala instalación del gestor de tareas **Celery** o de la cola de mensajes **RabbitMQ**.

Si el chequeo pasa a estado **Correcto**, entonces se habrá verificado el correcto funcionamiento del gestor de tareas y de la cola de mensajes. El siguiente paso es **comprobar las notificaciones**. Para ello, edite el chequeo recién creado y cambie el objetivo (*target*) a uno incorrecto, por ejemplo `uca.foobar` y espere tres minutos. En ese periodo de tiempo debería llegarle una notificación a través del correo electrónico. Si no es así, probablemente la causa esté en que no está bien configurada la cuenta de correo electrónico para el envío de notificaciones. Revise para ello la sección *Ejecución de los servicios*.

Si ha recibido correctamente la notificación a través del correo electrónico, entonces el sistema de mailing estará funcionando correctamente. El siguiente paso es repetir el proceso de **notificación con la aplicación Android**. Deberá instalar la aplicación en su móvil y acceder a la misma. Seguidamente, desde la aplicación web deberá editar de nuevo el chequeo y restaurar el objetivo original (`uca.es`), además de activar la notificación a través de Android.

Tras dos o tres minutos el sistema debería descubrir el cambio de estado y enviar las notificaciones, esta vez tanto por correo electrónico como por la aplicación Android. Si no recibe la notificación en su dispositivo móvil puede que haya algún

problema en la configuración de las claves para Google Cloud Messaging. De nuevo, debería revisar la sección *Ejecución de los servicios* del manual de instalación.

Si ha recibido la notificación correctamente, puede dar por terminada y **correcta** la verificación de la instalación.

A.3. Problemas habituales

A.3.1. Activación de nginx

A menudo, al instalar la configuración de nginx y reiniciar el servicio aparece el siguiente error:

```
nginx: [emerg] could not build the server_names_hash, you should increase server_names_h
```

Al parecer, nginx está configurado por defecto para usar subdominios de 32 caracteres como máximo. Si su dominio tiene más caracteres, deberá editar el fichero `/etc/nginx/nginx.conf` y descomentar la siguiente línea dentro del bloque `http`:

```
server_names_hash_bucket_size 64;
```

Tras eso, reinicie el servicio y el problema debería desaparecer.

A.3.2. Mejora del rendimiento

La configuración por defecto de **RabbitMQ** hace que consuma mucha memoria y puede llegar a resultar un problema en sistemas con recursos limitados, como los VPS de bajo coste.

Para arreglar esto, una opción es modificar el fichero de configuración, situado en `/etc/rabbitmq/rabbitmq.config` para limitar la memoria máxima usada por Rabbit a un 15 % del total. El contenido a añadir al fichero es el siguiente:

```
[
  {rabbit, [{vm_memory_high_watermark, 0.15}]}
].
```

Por otro lado, si el número de chequeos que se preveen tener es bajo, es posible reducir el número de workers que utiliza **Celery** modificando el fichero `supervisor-siteup.conf` que se generó en los pasos anteriores, cambiando el valor del parámetro `-c 16` a un número menor.

Apéndice B

Manual de usuario

En este capítulo se explicarán en detalle las instrucciones de uso de la plataforma web.

B.1. Introducción a SiteUp

SiteUp es un servicio web creado para la monitorización de servicios de Internet, especialmente pensado para dueños de sitios web y servidores que deseen estar informados en caso de cualquier eventualidad.

El uso de Internet en todos los ámbitos de la vida, desde las interacciones sociales hasta la búsqueda de empleo, han dado lugar a la creación de servicios accesibles a través de Internet y a nuevos modelos de negocio basados en aquellos. Resulta por tanto imprescindible contar con sistemas para monitorizar la disponibilidad de estos servicios y, en caso necesario, actuar de manera que puedan subsanarse las causas de los problemas detectados.

SiteUp cubre precisamente estas necesidades, permitiendo a los usuarios:

- Crear chequeos mediante envío de paquetes ICMP (Ping), para revisar que un servidor está activo y conectado a la red.
- Crear chequeos de registros DNS, de forma que el contenido de estos registros sea el correcto en todo momento.
- Crear chequeos de puertos remotos, para vigilar que los servicios concretos dentro de un servidor son accesibles.
- Crear chequeos mediante peticiones HTTP, para controlar que un servicio web responde de forma correcta y con el contenido adecuado.
- Establecer las opciones de periodicidad de estos chequeos, de acuerdo a las necesidades particulares de cada caso.

- Recibir notificaciones tanto a través de correo electrónico como mediante una aplicación móvil para el sistema operativo Android.

B.2. Requisitos previos

Para el uso correcto de la plataforma web SiteUp, el principal requisito es contar con un dispositivo con acceso a Internet que disponga de un navegador reciente, como por ejemplo Google Chrome. Cualquier tipo de tablet, teléfono móvil u ordenador personal podrá acceder a la plataforma web SiteUp, pero para aprovechar mejor el potencial del sistema se recomienda usar una computadora personal con las siguientes características:

- Navegador reciente. Se recomienda el uso de Google Chrome.
- Pantalla con resolución mínima de 1280 píxeles en su lado longitudinal.
- Conexión a Internet con un caudal de descarga de al menos 512kb/s.

Si desea hacer uso de la aplicación móvil, será necesario que el usuario cuente con un dispositivo con las siguientes características mínimas:

- Sistema operativo Android, versión mínima 2.3, versión recomendada 4.3.
- Pantalla de al menos 3 pulgadas de diagonal y resolución mínima de 320x480 píxeles.

B.3. Uso de la plataforma web

B.3.1. Acceso

A fecha de abril de 2014, la dirección de acceso a la plataforma web de SiteUp es <http://siteup.josetomastocino.com>. Simplemente introduzca esa dirección en su navegador y automáticamente será redirigido a la plataforma web, donde verá la pantalla de bienvenida.

B.3.2. Registro de usuario

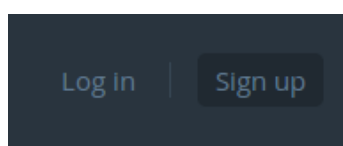


Figura B.1: Botones principales para el usuario anónimo

Si es la primera vez que accede al sitio web deberá registrarse. Para ello, debe pulsar en el botón **Sign up**, que se ve en la figura B.1.

Tras ello, aparecerá el formulario de registro de usuario, que se ve en la figura B.2

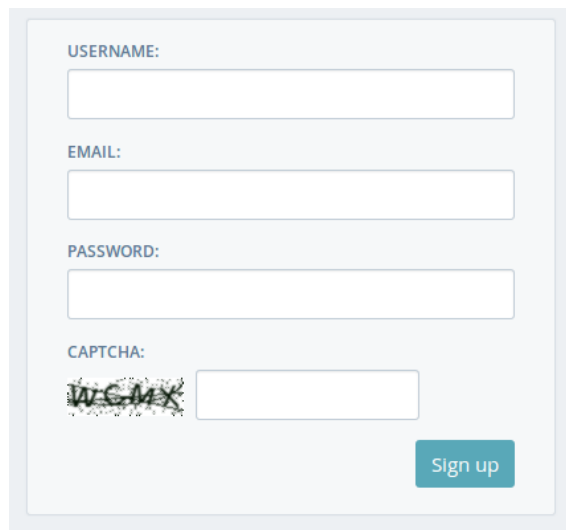
El formulario de registro de usuario está contenido en un recuadro gris claro. Incluye cuatro campos de entrada: 'USERNAME:', 'EMAIL:', 'PASSWORD:' y 'CAPTCHA:'. El campo 'CAPTCHA:' contiene una imagen de un código de verificación con los caracteres 'W C M X' y un campo de texto adyacente para introducirlos. En la esquina inferior derecha del recuadro hay un botón azul con el texto 'Sign up'.

Figura B.2: Formulario de registro de usuario

En el formulario deberá introducir los siguientes datos:

- **Nombre de usuario.** Debe tener 30 caracteres como máximo, y solo pueden usarse letras, números y guiones.
- **Dirección de correo electrónico.** Asegúrese de que es una dirección de correo correcta.
- **Contraseña.**
- **Captcha,** usado para evitar el registro automático de bots. Deberá introducir los caracteres que se ven en la imagen. Si no consigue ver bien los caracteres, actualice la página y aparecerá una nueva imagen.

Tras introducir los datos, deberá pulsar el botón **Sign up** del formulario. Si el registro es correcto, el sistema dará de alta su usuario y mostrará la pantalla B.3. Si no, volverá a aparecer el formulario con información sobre los errores en los datos.

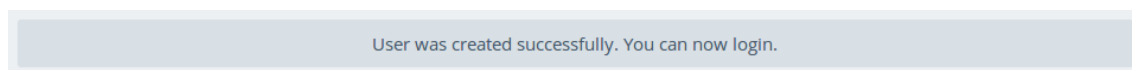


Figura B.3: Mensaje indicando el registro correcto

B.3.3. Inicio de sesión

Una vez que haya registrado su usuario, deberá pulsar en el botón **Log in**, como aparece en la imagen B.1. Esto le llevará al formulario de inicio de sesión que

aparece en la figura B.4

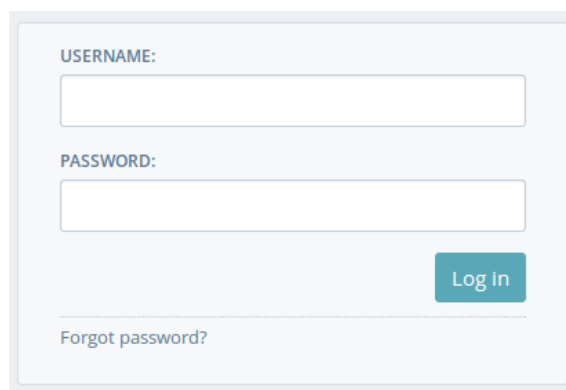
El formulario de inicio de sesión está contenido en un recuadro gris claro. En la parte superior, la etiqueta 'USERNAME:' precede a un campo de entrada rectangular. Debajo, la etiqueta 'PASSWORD:' precede a otro campo de entrada rectangular. A la derecha de estos campos, hay un botón rectangular de color azul con el texto 'Log In' en blanco. En la parte inferior izquierda del recuadro, hay un enlace hipervinculado que dice 'Forgot password?'.

Figura B.4: Formulario de inicio de sesión

Deberá introducir su nombre de usuario y la contraseña que escribió al registrarse, y posteriormente pulsar en el botón **Log in** del formulario. Si los datos son correctos, será llevado al panel principal de la web, donde podrá gestionar sus chequeos. Si no son correctos, volverá a aparecer el formulario, con información sobre los errores encontrados.

B.3.4. Recuperación de contraseña

Si se registró como usuario pero no recuerda su contraseña es posible recuperarla.

Para ello, vaya al formulario de inicio de sesión como se ha explicado en el paso anterior. En ese formulario, en la zona inferior, verá un enlace titulado **Forgot password?** que deberá pulsar. Después de hacerlo se mostrará el formulario de recuperación de contraseña, ilustrado en la figura B.5.

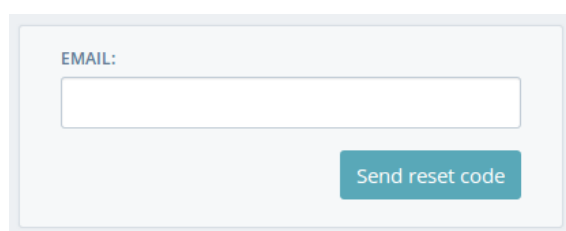
El formulario de recuperación de contraseña está contenido en un recuadro gris claro. En la parte superior, la etiqueta 'EMAIL:' precede a un campo de entrada rectangular. A la derecha de este campo, hay un botón rectangular de color azul con el texto 'Send reset code' en blanco.

Figura B.5: Formulario de recuperación de contraseña

En este formulario deberá introducir la dirección de correo electrónico que utilizó al registrarse, y pulsar el botón **Send reset code**. Si lo hace correctamente, aparecerá la pantalla de confirmación que se ve en la figura B.6.

El sistema habrá enviado un enlace a su dirección de correo electrónico. Deberá pulsar el enlace que aparece en ese correo. Asegúrese de que recibe correctamente el correo. Si no lo recibe, revise la carpeta de correo no deseado. El enlace del correo le llevará a un formulario de reseteo de contraseña como el que aparece en la figura B.7.

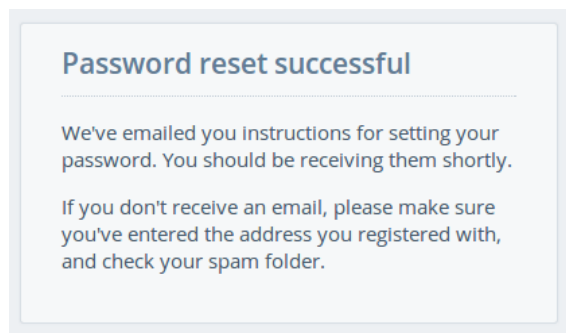


Figura B.6: Pantalla de confirmación de envío de código de reinicio

A screenshot of a web form for setting a new password. It contains two input fields: 'NEW PASSWORD:' and 'NEW PASSWORD CONFIRMATION:'. Below the second field is a blue button labeled 'Set new password'.

Figura B.7: Pantalla para la introducción de la nueva contraseña

En este formulario deberá escribir la nueva contraseña de su cuenta dos veces, para confirmar que la escribe correctamente. Tras ello, deberá pulsar el botón **Set new password**, lo cual confirmará los cambios.

Tras este proceso, podrá iniciar sesión como se ha descrito previamente, esta vez utilizando la nueva contraseña.

B.3.5. Pantalla de listado de chequeos

En la figura B.8 se puede ver el aspecto del panel de control de la plataforma web SiteUp. Desde este panel podrá hacer una gran cantidad de acciones, desde crear grupos de chequeos hasta editar sus detalles de usuario.

Es posible acceder a esta pantalla desde cualquier sitio de la web si está logueado, pulsando en el botón **Dashboard** de la cabecera de la web, tal y como se ve en la figura B.9.

B.3.6. Cerrar sesión

Si ha iniciado sesión y desea cerrarla, debe dirigirse a los botones principales, situados en el lado derecho de la cabecera de la web, tal y como aparecen en la

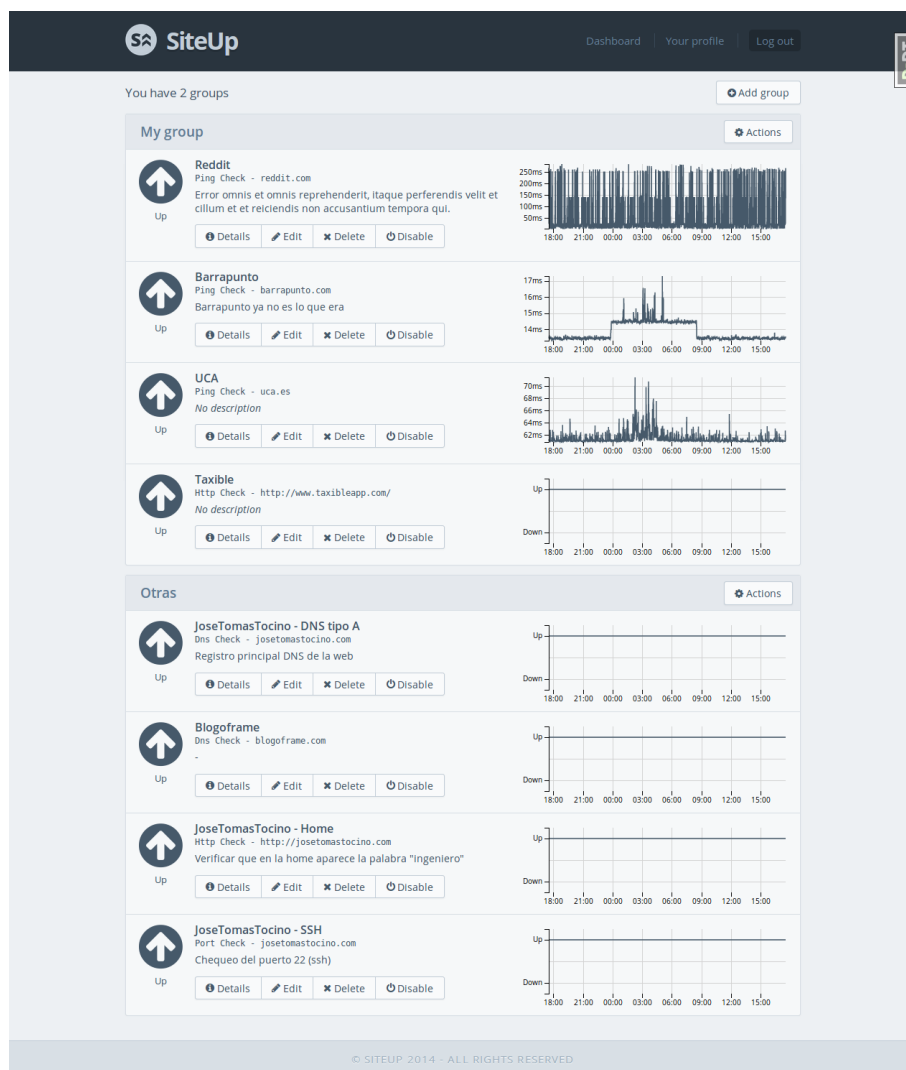


Figura B.8: Panel de control

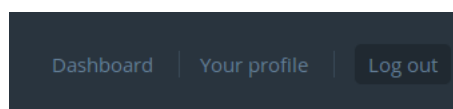


Figura B.9: Botones principales para el usuario logueado

figura B.9. Deberá pulsar el botón **Log out**, tras lo cual se cerrará su sesión y será redirigido a la pantalla inicial de SiteUp.

B.3.7. Edición de datos de usuario

Si desea editar alguno de los datos de usuario que ingresó cuando hizo el registro, deberá primero pulsar el botón **Your profile**, situado en la cabecera de la web. Es el segundo de los botones que aparecen, tal y como se ve en la figura B.9.

Una vez hecho esto, aparecerá el formulario de edición de usuario, ilustrado en la figura B.10, en el que verá sus actuales datos de usuario.

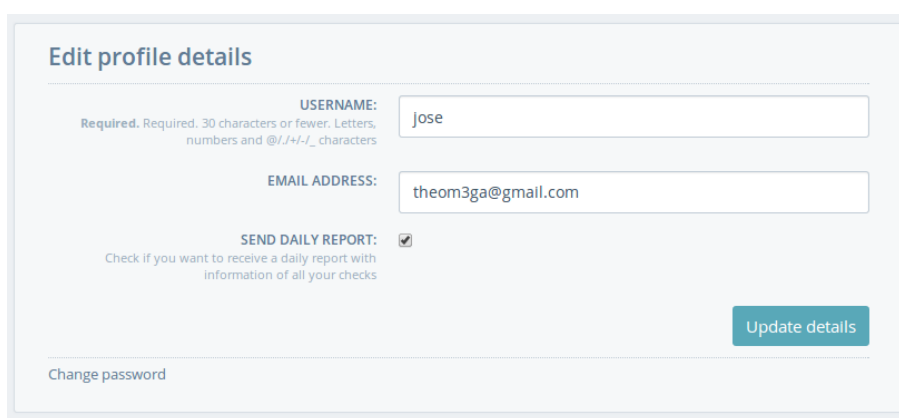


Figura B.10: Pantalla de edición del perfil de usuario

En este formulario podrá editar:

- Su **nombre** de usuario.
- Su **dirección** de correo electrónico.
- Si desea o no recibir **resúmenes diarios** sobre el estado de sus chequeos.

Una vez editados los datos que desee deberá pulsar el botón **Update details**. También podrá editar su contraseña pulsando el botón **Change password**, situado al pie del formulario a la izquierda.

B.3.8. Creación de grupo de chequeos

En SiteUp, los chequeos se organizan en **grupos de chequeos**, lo que permite trabajar con ellos más fácilmente a la hora de activarlos, desactivarlos o borrarlos.

Para crear un grupo de chequeos deberá pulsar el botón **Add group**, situado en la zona superior derecha del panel principal. El botón puede verse en la figura B.11.

Tras pulsar el botón aparecerá el formulario de creación de nuevo grupo. En él deberá introducir los datos del nuevo grupo. En particular:

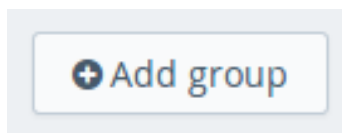


Figura B.11: Botón para crear nuevo grupo de chequeos

- El **título** que identifica al grupo.

Tras ello, deberá pulsar el botón **Create group**, y el grupo será creado

Un formulario con el título "Create group of checks" en azul. Debajo, hay un campo de entrada con el label "TITLE:" y el texto "Required. Title for the group of checks" en gris. A la derecha del campo, hay un botón azul con el texto "Create group".

Figura B.12: Formulario de creación de grupo de chequeos

B.3.9. Operaciones con grupos de chequeos

En la figura B.13 se puede ver el menú de operaciones disponibles para un grupo de chequeos. Este menú aparece en el panel principal, al lado derecho junto al título de cada grupo de chequeos.

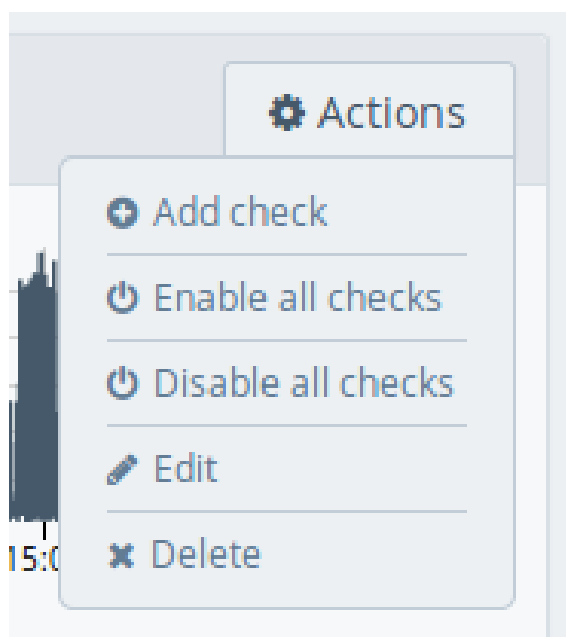


Figura B.13: Menú de acciones de grupo

Edición de grupo de chequeos

En el menú de operaciones del grupo de chequeos, pulse el botón **Edit**. Aparecerá un formulario similar al de creación de grupo, en el que podrá editar el título del grupo de chequeos.

Borrado de grupo de chequeos

Para borrar un grupo de chequeos, pulse el botón **Delete** en el menú de operaciones del grupo. Aparecerá un formulario como el de la figura B.14, pidiéndole confirmación del borrado.

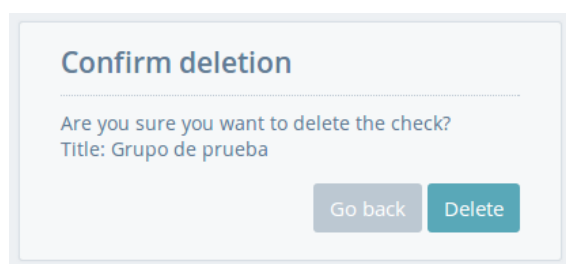
El formulario tiene un fondo gris claro. En la parte superior, el título "Confirm deletion" está en azul. Debajo, una línea de puntos separa el título de la pregunta "Are you sure you want to delete the check?" en azul. A continuación, se muestra "Title: Grupo de prueba" en gris. En la parte inferior derecha, hay dos botones: "Go back" (gris) y "Delete" (azul).

Figura B.14: Formulario de confirmación de borrado

Si pulsa en el botón **Go back**, volverá al panel principal, cancelando el borrado. Si pulsa el botón **Delete**, confirmará la operación y el sistema borrará el grupo de chequeos, así como todos los chequeos que contenga.

Activación y desactivación de grupo de chequeos

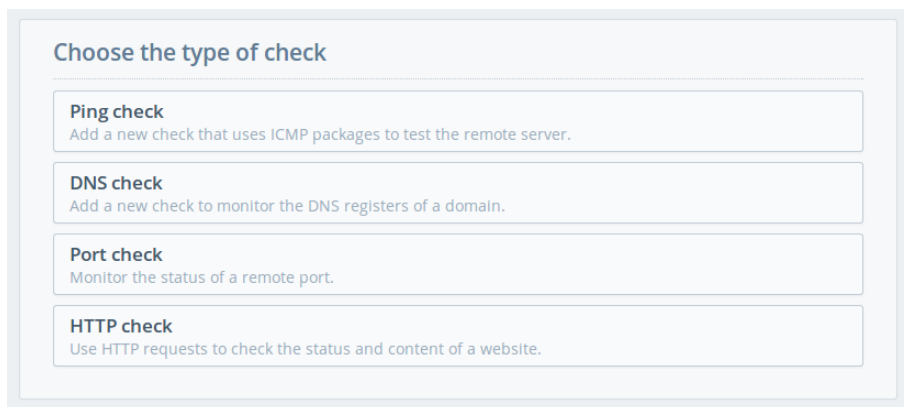
Es posible activar y desactivar de forma grupal todos los chequeos de un grupo. Para ello, deberá dirigirse al menú de operaciones del grupo y pulsar en **Enable all checks** o en **Disable all checks**, según sea su decisión.

B.3.10. Creación de chequeo

Como se ha comentado previamente, el elemento principal de SiteUp es el **chequeo**. El usuario da de alta chequeos dentro de un grupo, y estos chequeos se lanzan periódicamente, generando datos.

Para crear un chequeo dentro de un grupo deberá ir al menú de operaciones del grupo, que aparece en la figura B.13 y se sitúa a la derecha del título de un grupo de chequeos. Dentro del menú, deberá pulsar la opción **Add check**.

Tras esto aparecerá una pantalla en la que deberá seleccionar el tipo de chequeo que desea crear. La pantalla es similar a la de la figura B.15. Como se explicó en la



Choose the type of check

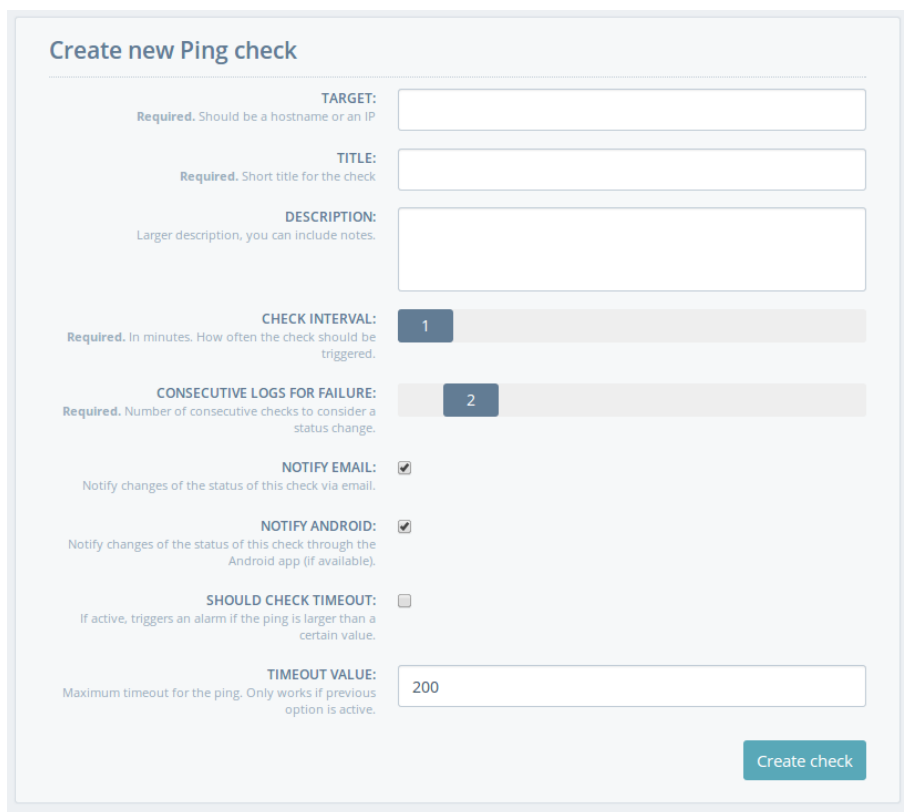
- Ping check**
Add a new check that uses ICMP packages to test the remote server.
- DNS check**
Add a new check to monitor the DNS registers of a domain.
- Port check**
Monitor the status of a remote port.
- HTTP check**
Use HTTP requests to check the status and content of a website.

Figura B.15: Pantalla de selección del tipo de chequeo a crear

introducción de este manual, es posible crear cuatro tipo de chequeos: chequeos de ping, chequeos de puerto, chequeos de registros DNS y chequeos por HTTP.

Si pulsa en el botón del tipo que desea crear, aparecerá un formulario con campos para rellenar.

Creación de un chequeo por ping



Create new Ping check

TARGET:
Required. Should be a hostname or an IP

TITLE:
Required. Short title for the check

DESCRIPTION:
Larger description, you can include notes.

CHECK INTERVAL:
Required. In minutes. How often the check should be triggered.

CONSECUTIVE LOGS FOR FAILURE:
Required. Number of consecutive checks to consider a status change.

NOTIFY EMAIL: ☒
Notify changes of the status of this check via email.

NOTIFY ANDROID: ☒
Notify changes of the status of this check through the Android app (if available).

SHOULD CHECK TIMEOUT: ☐
If active, triggers an alarm if the ping is larger than a certain value.

TIMEOUT VALUE:
Maximum timeout for the ping. Only works if previous option is active.

Create check

Figura B.16: Formulario de creación de chequeo por ping

El formulario para crear un chequeo de tipo ping aparece en la figura B.16. Los campos que deberá rellenar son los siguientes:

- **Target:** es el objetivo a chequear. Debe ser un *hostname* o una IP.
- **Título** del chequeo.
- **Descripción** opcional del chequeo.
- **Check interval:** intervalo entre chequeos, definido en minutos. El valor mínimo es de 1 minuto.
- **Consecutive logs for failure:** número de chequeos inválidos consecutivos necesarios para considerar un fallo en el servicio monitorizado.
- **Notify email:** indica si los cambios de estado en este chequeo deberán ser notificados por correo electrónico.
- **Notify Android:** indica si los cambios de estado en este chequeo deberán ser notificados mediante la aplicación Android.
- **Should check timeout:** indica si hay que verificar que el ping enviado vuelve dentro de un tiempo máximo determinado.
- **Timeout value:** si la anterior opción está activa, este campo define el número de milisegundos que puede tardar en responder el servidor al ping, como máximo.

Creación de un chequeo de puerto

El formulario para crear un chequeo de puertos aparece en la figura B.17. Los campos que deberá rellenar son los siguientes:

- **Título** del chequeo.
- **Descripción** opcional del chequeo.
- **Target:** es el objetivo a chequear. Debe ser un *hostname* o una IP.
- **Target port:** es el puerto a chequear. Debe ser un entero.
- **Check for string:** opcionalmente, se puede indicar una cadena que debe estar presente en la respuesta al conectar con el puerto.
- **Check interval:** intervalo entre chequeos, definido en minutos. El valor mínimo es de 1 minuto.
- **Consecutive logs for failure:** número de chequeos inválidos consecutivos necesarios para considerar un fallo en el servicio monitorizado.
- **Notify email:** indica si los cambios de estado en este chequeo deberán ser notificados por correo electrónico.

Create new Port check

TITLE:
Required. Short title for the check

DESCRIPTION:
Larger description, you can include notes.

TARGET:
Required. Should be a hostname or an IP

TARGET PORT:
Required. Remote network port to check.
Click to see common ports

CHECK FOR STRING:
Optionally, you can check if the response contains a certain string.

CHECK INTERVAL:
Required. In minutes. How often the check should be triggered. 1

CONSECUTIVE LOGS FOR FAILURE:
Required. Number of consecutive checks to consider a status change. 2

NOTIFY EMAIL: ☒
Notify changes of the status of this check via email.

NOTIFY ANDROID: ☒
Notify changes of the status of this check through the Android app (if available).

Create check

Figura B.17: Formulario de creación de chequeo de puerto

- **Notify Android:** indica si los cambios de estado en este chequeo deberán ser notificados mediante la aplicación Android.

Creación de un chequeo de DNS

El formulario para crear un chequeo de registro DNS aparece en la figura B.18. Los campos que deberá rellenar son los siguientes:

- **Título** del chequeo.
- **Descripción** opcional del chequeo.
- **Target:** es el objetivo a chequear. Debe ser un *hostname*.
- **Record type:** indica el tipo de registro a chequear. Puede ser A, AAAA, CNAME, TXT y MX.
- **Resolved address:** indica el contenido que debe tener el registro.
- **Check interval:** intervalo entre chequeos, definido en minutos. El valor mínimo es de 1 minuto.
- **Consecutive logs for failure:** número de chequeos inválidos consecutivos necesarios para considerar un fallo en el servicio monitorizado.

Create new Dns check

TITLE:
Required. Short title for the check

DESCRIPTION:
Larger description, you can include notes.

TARGET:
Required. Should be a hostname

RECORD TYPE:
Required. Type of dns resource record
A

RESOLVED ADDRESS:
Required. Should be a valid value for the selected record type

CHECK INTERVAL:
Required. In minutes. How often the check should be triggered.
1

CONSECUTIVE LOGS FOR FAILURE:
Required. Number of consecutive checks to consider a status change.
2

NOTIFY EMAIL: ☒
Notify changes of the status of this check via email.

NOTIFY ANDROID: ☒
Notify changes of the status of this check through the Android app (if available).

Create check

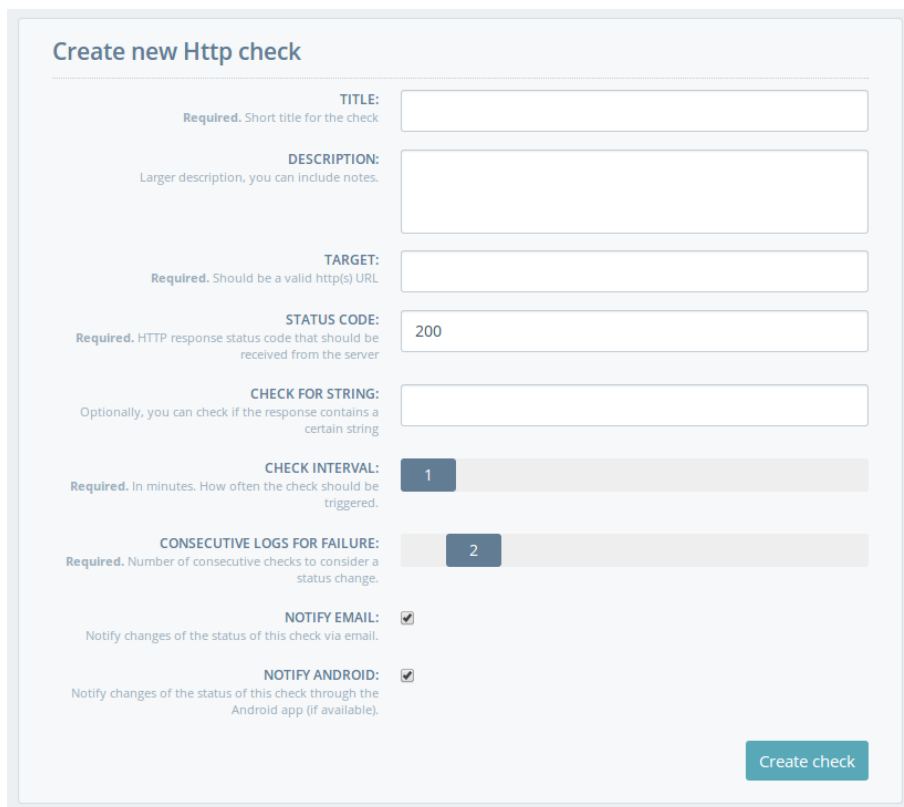
Figura B.18: Formulario de creación de chequeo de registro DNS

- **Notify email:** indica si los cambios de estado en este chequeo deberán ser notificados por correo electrónico.
- **Notify Android:** indica si los cambios de estado en este chequeo deberán ser notificados mediante la aplicación Android.

Creación de un chequeo por HTTP

El formulario para crear un chequeo por HTTP aparece en la figura B.19. Los campos que deberá rellenar son los siguientes:

- **Título** del chequeo.
- **Descripción** opcional del chequeo.
- **Target:** es el objetivo a chequear. Debe ser una URL.
- **Status code:** es el código de estado que deberá devolver la petición.
- **Check for string:** opcionalmente, se puede indicar una cadena que debe estar presente en la respuesta a la petición.
- **Check interval:** intervalo entre chequeos, definido en minutos. El valor mínimo es de 1 minuto.



Create new Http check

TITLE:
Required. Short title for the check

DESCRIPTION:
Larger description, you can include notes.

TARGET:
Required. Should be a valid http(s) URL

STATUS CODE:
Required. HTTP response status code that should be received from the server

CHECK FOR STRING:
Optionally, you can check if the response contains a certain string

CHECK INTERVAL:
Required. In minutes. How often the check should be triggered.

CONSECUTIVE LOGS FOR FAILURE:
Required. Number of consecutive checks to consider a status change.

NOTIFY EMAIL: ☒
Notify changes of the status of this check via email.

NOTIFY ANDROID: ☒
Notify changes of the status of this check through the Android app (if available).

Create check

Figura B.19: Formulario de creación de chequeo por HTTP

- **Consecutive logs for failure:** número de chequeos inválidos consecutivos necesarios para considerar un fallo en el servicio monitorizado.
- **Notify email:** indica si los cambios de estado en este chequeo deberán ser notificados por correo electrónico.
- **Notify Android:** indica si los cambios de estado en este chequeo deberán ser notificados mediante la aplicación Android.

B.3.11. Operaciones con chequeos

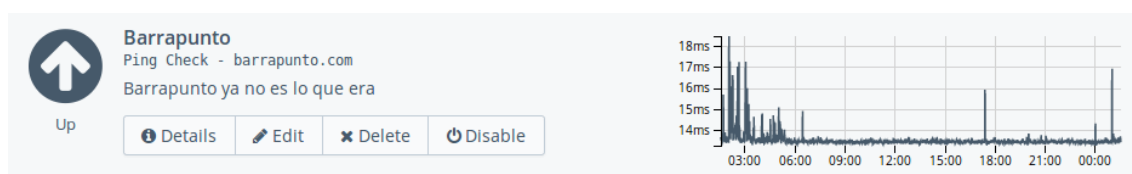


Figura B.20: Aspecto de un chequeo válido

Una vez creado un chequeo, aparecerá en el panel de control con el aspecto de la figura B.20. Como se aprecia en la figura, se muestra mediante un icono el estado del chequeo, los datos (título, tipo, descripción) y una gráfica que ilustra el estado del chequeo en las últimas 24 horas.

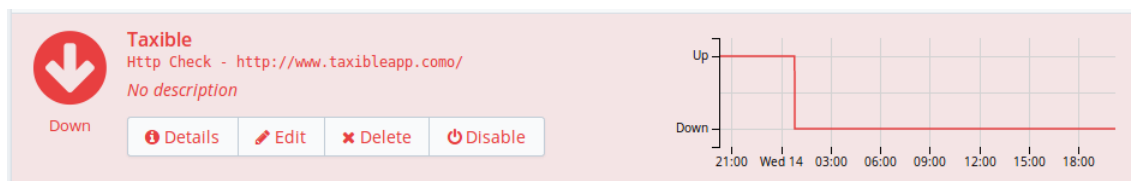


Figura B.21: Aspecto de un chequeo en estado de error

Si un chequeo detecta un error, cambia de apariencia, pasando a tener el aspecto que ofrece la figura B.21. Además, según se haya configurado en las opciones del chequeo, se envía una notificación al usuario informando del cambio de estado del chequeo.

Cuando se ve en la lista general de chequeos, cada chequeo tiene una serie de botones que permiten llevar a cabo operaciones diversas, que se detallan a continuación.

Activación y desactivación de un chequeo

Para desactivar un chequeo, tiene que pulsar el botón **Disable** de la botonera, que solo aparecerá si el chequeo está activado. De igual modo, si el chequeo está desactivado, el botón que aparecerá tendrá la etiqueta **Enable**, que podrá pulsar para activarlo.

Borrado de un chequeo

En la botonera se sitúa el botón **Delete**, que podrá pulsar para borrar el chequeo. Aparecerá un formulario pidiéndole confirmación. Si confirma la acción, se procederá al borrado del chequeo y todos los datos dependientes de éste.

Edición de un chequeo

Si desea editar un chequeo debe pulsar el botón **Edit**, situado en la botonera del chequeo. Aparecerá un formulario similar al de creación de chequeo, pero con los datos actuales ya introducidos. Puede modificar los datos que desee y confirmar los cambios pulsando el botón **Update check** del formulario.

Vista de detalle del chequeo

En la botonera del chequeo, en el panel de control, verá un botón **Details**. Si lo pulsa será llevado a la pantalla de vista de detalle del chequeo, que se ilustra en la figura B.22. Esta pantalla ofrece mucha información sobre el chequeo. En particular:

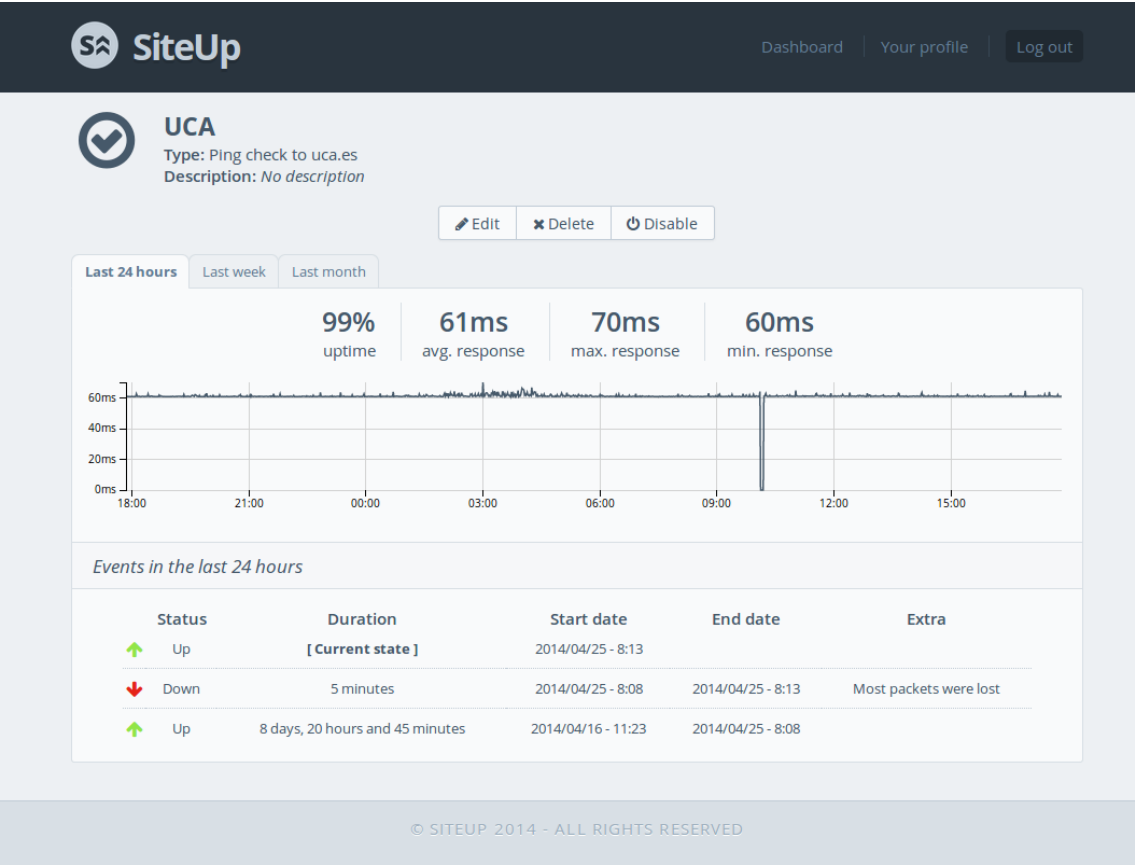


Figura B.22: Pantalla de detalle de chequeo

- En la parte superior se sitúa la información general:
 - Icono con el estado del chequeo.
 - Título.
 - Tipo de chequeo.
 - Descripción.
 - Botonera de acciones.
- En la parte media de la pantalla se sitúa un bloque con pestañas. Mediante estas pestañas es posible indicar sobre qué periodo de tiempo queremos visualizar los datos: últimas 24 horas, última semana, último mes, etc. La información disponible es la siguiente:
 - Porcentaje de *uptime* del chequeo.
 - (*Solo chequeos tipo ping*) Tiempo de respuesta mínimo, máximo y promedio.
 - Gráfica con el estado del chequeo.
 - Eventos durante ese periodo de tiempo. Un evento es un cambio de estado del chequeo, con la siguiente información:
 - Tipo de cambio de estado (*Up, Down*).
 - Duración del estado.
 - Hora de comienzo del estado.
 - Hora de fin del estado. Si es el estado en el que el chequeo se encuentra actualmente, no aparecerá ninguna fecha.
 - Extra: muestra información adicional para estados de error.

B.3.12. Recepción de notificaciones por correo electrónico

Notificaciones de cambio de estado

Cuando el sistema detecte un cambio de estado en un chequeo, y esté activada la notificación por correo electrónico, se enviará un email como el que aparece en la figura B.23.

En ese email se le indicará el título del chequeo, el nuevo estado, la fecha y hora en la que ha pasado al nuevo estado, y un enlace directo al panel de detalle del chequeo en SiteUp, de forma que pueda obtener más información.

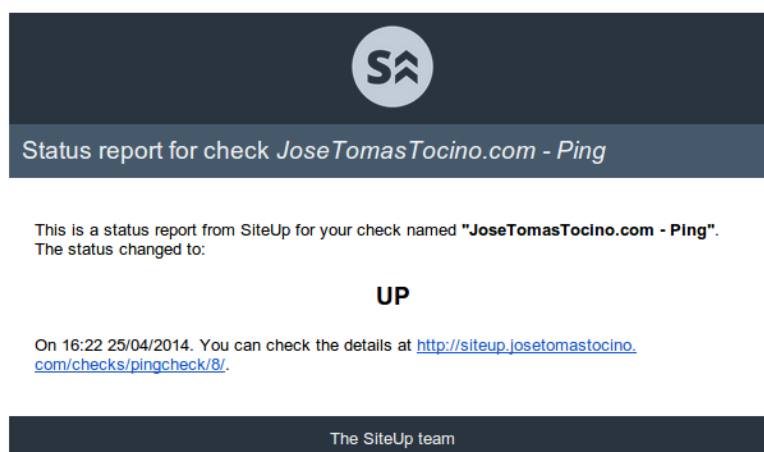


Figura B.23: Email de notificación de cambio de estado

Notificaciones de resumen diarias

Si ha activado el envío de resúmenes diarios sobre el estado de sus chequeos, el sistema enviará todos los días un correo electrónico con información. El aspecto del correo que recibirá se puede ver en la figura B.24

B.4. Uso de la aplicación móvil

La aplicación móvil **SiteUp Client** es una aplicación para para el sistema operativo Android que permite a sus usuarios revisar los chequeos que tiene dados de alta en la plataforma web SiteUp, así como recibir de manera instantánea notificaciones *push* sobre el estado de sus chequeos.

B.4.1. Instalación

La aplicación móvil se instala copiando y ejecutando el fichero APK en el dispositivo. En un futuro se podrá encontrar en la *Play Store* de Google.

B.4.2. Lanzamiento

En la figura B.25 se puede observar el aspecto de la aplicación **SiteUp Client** en el lanzador de Android. Para ejecutarla, simplemente pulse sobre el icono de la aplicación.

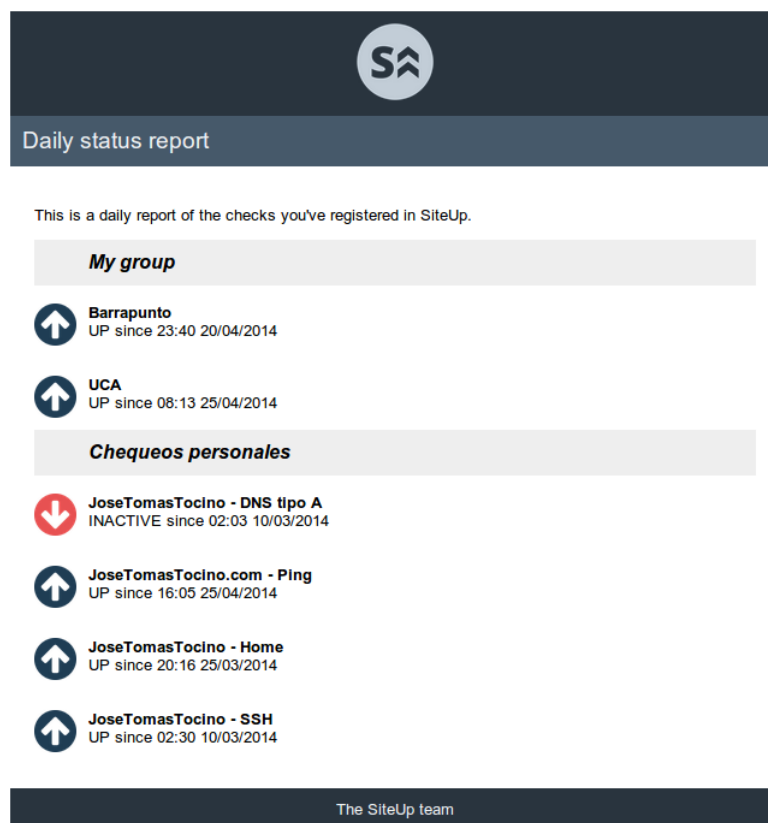


Figura B.24: Email diario de resumen

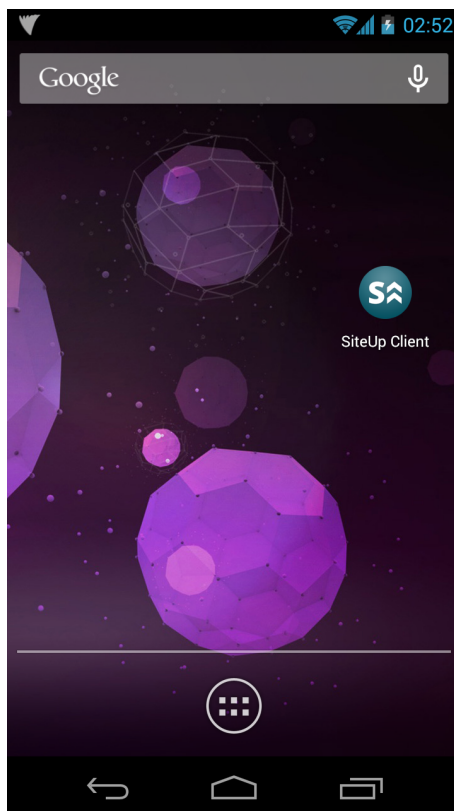


Figura B.25: Icono de SiteUp en el lanzador

B.4.3. Carga de la aplicación

Tras lanzar la aplicación, el sistema cargará los datos necesarios y se comunicará con los servidores de Google para obtener los datos necesarios. La apariencia de la aplicación durante este proceso se puede ver en la figura B.27.

B.4.4. Inicio de sesión

Tras la carga, la primera vez que lance la aplicación será necesario que inicie sesión con las credenciales que utilizó en la plataforma web. Para ello, tendrá que introducir su nombre de usuario y contraseña en el formulario que aparece en la figura B.26, y pulsar el botón **Sign in**.

La aplicación móvil se comunicará con los servidores centrales de SiteUp, validará sus credenciales de usuario y, si son correctas, le dará acceso al sistema. Si las credenciales de usuario no son correctas, el formulario mostrará un error.



Figura B.26: Pantalla de carga de la aplicación móvil

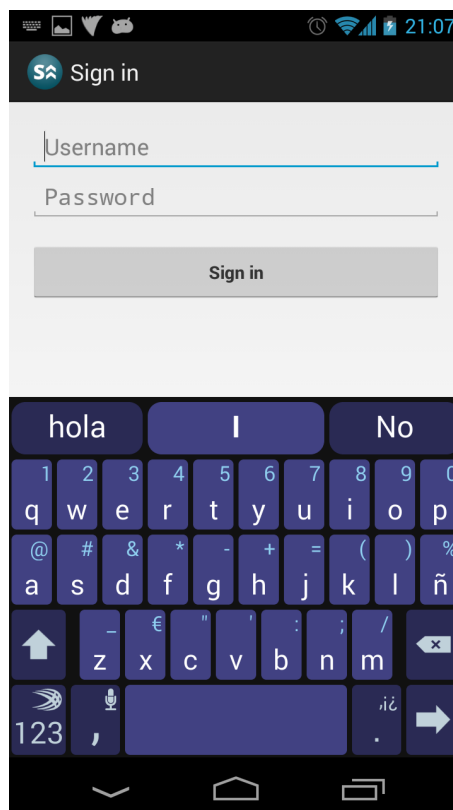


Figura B.27: Pantalla de inicio de sesión de la aplicación móvil

B.4.5. Revisión de chequeos

Si ha iniciado sesión correctamente, tendrá la opción de revisar la lista de chequeos que tiene dados de alta en la plataforma web **SiteUp**. La apariencia de la lista de chequeos es igual a la de la figura B.28.

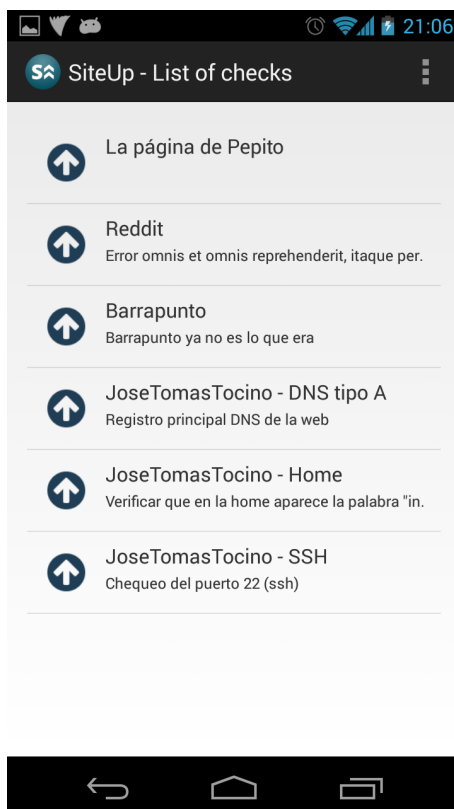


Figura B.28: Pantalla de chequeos de la aplicación móvil

Si pulsa en cualquiera de los chequeos, será redirigido a la plataforma web, que podrá navegar sin problema gracias a que se encuentra preparada para su uso en dispositivos móviles.

B.4.6. Recepción de notificaciones

La utilidad principal de la aplicación es la posibilidad de recibir notificaciones en cualquier momento provenientes del servidor – conocidas como notificaciones *push*.

El procedimiento es transparente al usuario. Simplemente lance la aplicación e inicie sesión. Tras hacer esto, la aplicación estará preparada para recibir notificaciones desde el servidor.

Cuando se reciba una notificación, la aplicación mostrará un mensaje en el área de notificaciones del móvil, con información resumida. Puede ver el aspecto de la

notificación en la figura B.29. Si pulsa en la notificación será llevado a los detalles del chequeo para obtener más detalles sobre el origen del aviso.

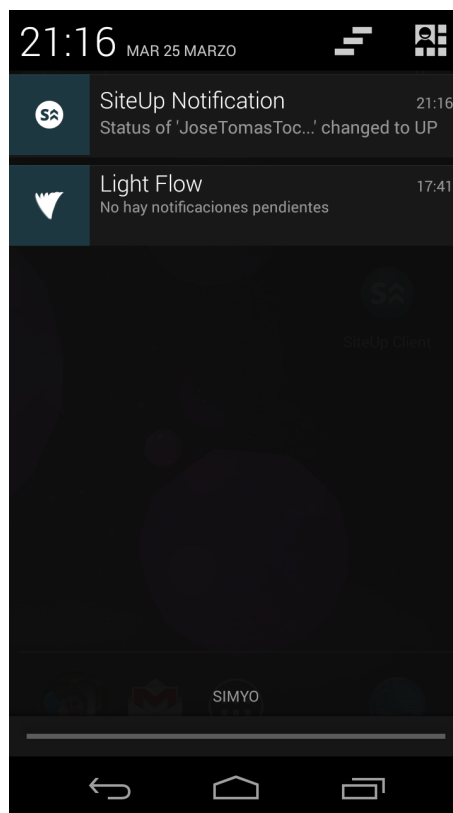


Figura B.29: Aspecto de las notificaciones

Apéndice C

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited

in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated

HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as

long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the

public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover

Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works

permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your

rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your

documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliografía y referencias

- [1] *Adobe Photoshop*. <http://www.adobe.com/es/products/photoshop.html>.
- [2] *Amazon just lost \$4.8M after going down for 40 minutes*. <http://www.geekwire.com/2013/amazon-lost-5m-40-minutes/>.
- [3] *Android Operating System*. <http://www.android.com/>.
- [4] *Android Studio IDE*. <http://developer.android.com/sdk/installing/studio.html>.
- [5] *Apple iOS 7*. <http://www.apple.com/es/ios/>.
- [6] *Boletín oficial del Estado núm 82 del 4 de abril de 2009*. URL <https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf>.
- [7] *Bottle: Python Web Framework*. <http://bottlepy.org/>.
- [8] *Cascading Style Sheets Snapshot 2010*. <http://www.w3.org/TR/CSS/>.
- [9] *Celery: Distributed Task Queue*. <http://www.celeryproject.org/>.
- [10] *Compass, an open-source CSS Authoring Framework*. <http://compass-style.org/>.
- [11] *ComScore Reports January 2014, US, Smartphone Subscriber Market Share*. http://www.comscore.com/Insights/Press_Releases/2014/3/comScore_Reports_January_2014_US_Smartphone_Subscriber_Market_Share.
- [12] *D3.js – Data-Driven Documents*. <http://d3js.org/>.
- [13] *Django Frequently Asked Questions*. URL <https://docs.djangoproject.com/en/dev/faq/general/>.
- [14] *Django, the web framework for perfectionists with deadlines*. <http://www.djangoproject.com>.
- [15] *draw.io, a free online diagram drawing application*. <http://draw.io>.
- [16] *Fabric*. <http://www.fabfile.org/>.
- [17] *Facebook to Acquire Whatsapp for \$19B*. <http://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/>.
- [18] *Flask, a microframework for Python*. <http://flask.pocoo.org/>.

- [19] *Git, a distributed version control system.* <http://git-scm.com/>.
- [20] *GNU Emacs.* <http://www.gnu.org/software/emacs/>.
- [21] *Google Apps for Business.* <http://www.google.com/enterprise/apps/business/>.
- [22] *Google Cloud Messaging for Android.* <http://developer.android.com/google/gcm/index.html>.
- [23] *Grunt: The JavaScript Task Runner.* <http://gruntjs.com/>.
- [24] *Hetzner vServer VQ7.* http://www.hetzner.de/en/hosting/produkte_vserver/vq7.
- [25] *HTML 5 Nightly Draft.* <http://www.w3.org/html/wg/drafts/html/master/Overview.html>.
- [26] *HttpBin, HTTP Client Testing Service.* <http://httpbin.org>.
- [27] *JoseTomasTocino's GitHub - Forja de SiteUp.* <https://github.com/JoseTomasTocino/pfc-ii>.
- [28] *jQuery – write less, do more.* <http://jquery.com/>.
- [29] *Linux Programmer's Manual - Overview of Linux Capabilities.* URL <http://man7.org/linux/man-pages/man7/capabilities.7.html>.
- [30] *Listado de Premios Locales del VIII CUSL.* URL <http://osl.uca.es/?q=node/1516>.
- [31] *MySQL, The world's most popular open source database.* <http://mysql.com>.
- [32] *Nginx web server.* <http://nginx.org>.
- [33] *node.js.* <http://nodejs.org/>.
- [34] *Open Sans.* <http://opensans.com/>.
- [35] *PEP 8 - Style Guide for Python Code.* URL <http://legacy.python.org/dev/peps/pep-0008/>.
- [36] *pip, a tool for installing Python packages.* <http://www.pip-installer.org/>.
- [37] *Python, the official website.* <https://www.python.org/>.
- [38] *Sass: Syntactically Awesome Style Sheets.* <http://sass-lang.com/>.
- [39] *South.* <http://south.aeracode.org/>.
- [40] *SQLite.* <https://sqlite.org/>.
- [41] *Start Developing iOS applications.* <https://developer.apple.com/library/iOS/referencelibrary/GettingStarted/RoadMapiOS/index.html>.
- [42] *Sublime Text: The text editor you'll fall in love with.* <http://sublimetext.com>.

- [43] VIII Concurso Universitario de Software Libre. URL <http://concursosoftwarelibre.org>.
- [44] VirtualEnv, a tool to create isolated Python environments. <http://www.virtualenv.org/>.
- [45] VirtualEnvWrapper, a set of extensions to VirtualEnv. <http://virtualenvwrapper.readthedocs.org/>.
- [46] Why IDC predicts Windows Phone will surpass iOS by 2016. <http://www.wired.com/2012/06/why-idc-predicts-windows-phone-will-surpass-ios-by-2016/>.
- [47] Why Startups Fail - Business Model Failure. URL <http://www.forentrepreneurs.com/business-models/why-startups-fail/>.
- [48] Peter Salus. *A quarter century of UNIX*. Addison-Wesley Pub. Co, Reading, Mass, 1994. ISBN 0201547775.
- [49] Wikipedia. *Ataque Man-in-the-middle* — Wikipedia, La enciclopedia libre, 2013. URL http://es.wikipedia.org/w/index.php?title=Ataque_Man-in-the-middle&oldid=69452204, [Internet; descargado 29-abril-2014].
- [50] Wikipedia. *Defacement* — Wikipedia, La enciclopedia libre, 2013. URL <http://es.wikipedia.org/w/index.php?title=Defacement&oldid=65253856>, [Internet; descargado 28-abril-2014].
- [51] Wikipedia. *Cron* — Wikipedia, The Free Encyclopedia, 2014. URL <http://en.wikipedia.org/w/index.php?title=Cron&oldid=604984586>, [Online; accessed 22-April-2014].
- [52] Wikipedia. *Denial-of-service attack* — Wikipedia, The Free Encyclopedia, 2014. URL http://en.wikipedia.org/w/index.php?title=Denial-of-service_attack&oldid=604626201, [Online; accessed 22-April-2014].
- [53] Wikipedia. *Inyección SQL* — Wikipedia, La enciclopedia libre, 2014. URL http://es.wikipedia.org/w/index.php?title=Inyecci%C3%B3n_SQL&oldid=74294994, [Internet; descargado 13-mayo-2014].
- [54] Wikipedia. *JavaScript* — Wikipedia, The Free Encyclopedia, 2014. URL <http://en.wikipedia.org/w/index.php?title=JavaScript&oldid=605304642>, [Online; accessed 22-April-2014].